Vers des générateurs de nombres aléatoires uniformes et gaussiens à très haut débit

Renaud Santoro

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Traitement du Signal et Télécommunications

École doctorale Matisse

présentée par

Renaud SANTORO

préparée à l'unité de recherche UMR6074 IRISA
Institut de recherche en informatique et systèmes aléatoires - CAIRN
École Nationale Supérieure des Sciences Appliquées et de Technologie

---

Vers des générateurs de nombres aléatoires uniformes et gaussiens à très haut débit

Thèse soutenue à Lannion
le Jeudi 17 Décembre 2009

devant le jury composé de :

Viktor FISCHER
Professeur des Universités
Université Jean Monnet / président

Bruno ROUZEYRE
Professeur des Universités
Université de Montpellier / rapporteur

Russell TESSIER
Professeur
University of Massachusetts / rapporteur

Paul FORTIER
Professeur
Université Laval / examinateur

Olivier SENTIEYS
Professeur des Universités
Université de Rennes 1 / directeur de thèse

Sébastien ROY
Professeur
Université Laval / co-directeur de thèse

Remerciements

Je remercie tout particulièrement Monsieur Olivier Sentieys, Professeur des Universités à l'ENSSAT et responsable de l'équipe CAIRN, pour m'avoir accueilli dans son équipe de recherche et pour avoir accepter de diriger cette thèse. Je tiens à remercier également Monsieur Sébastien Roy, Professeur adjoint à l'Université Laval de Québec pour avoir accepté le co-encadrement de cette thèse et m'avoir accueilli dans son équipe de recherche pendant un an. Qu'ils soient assurés de ma gratitude.

Je tiens également à remercier Monsieur Viktor Fischer pour avoir accepté de présider le jury de soutenance, Messieurs Bruno Rouzeyre et Russell Tessier pour avoir accepté de rapporter sur ce manuscrit et enfin Monsieur Paul Fortier pour avoir accepté d'examiner ces travaux.

Je tiens également à remercier Arnaud Tisserand pour ses conseils, sa rigueur et son ouverture scientifique. Merci encore à Thomas Anger, Arnaud Carer, Rémi Pallas et Philippe Quéméré pour leur contribution dans le développement des circuits ASIC OCHRE.

Que tous les membres de l'équipe CAIRN et le personnel de l'ENSSAT acceptent mes remerciements pour avoir contribué à rendre cette expérience enrichissante aussi bien sur le plan professionnel que personnel.

Enfin, mon dernier remerciement, mais non des moindres, à Caroline, pour avoir su me supporter durant ces trois dernières années.

Table des matières

Introduction **1**

Plan du mémoire 5

I État de l'art **7**

1 Introduction 7

2 Rôle et principe de fonctionnement 7

2.1 Définitions préliminaires 7

2.2 Applications 8

2.2-1 Cryptographie 9

2.2-2 Méthodes de Monte-Carlo 14

2.2-3 Communications numériques 15

3 Générateurs de nombres vraiment aléatoires 16

3.1 TRNG basés sur l'amplification d'un bruit physique 17

3.1-1 TRNG proposé par Bagini et al. 17

3.1-2 TRNG proposé par Jun et al. 17

3.2 TRNG basés sur l'échantillonnage d'oscillateurs 18

3.2-1 TRNG proposé par Tsoi et al. 18

3.2-2 TRNG proposé par Fischer et al. 19

3.2-3 TRNG proposé par Bucci et al. 23

3.2-4 TRNG proposé par Kohlbrenner et al. 24

3.2-5 TRNG proposé par Tkacik et al. 27
3.2-6 TRNG proposé par Sunar et al. 28
3.2-7 Améliorations proposées par Schellekens et al. 29
3.2-8 Améliorations proposées par Yoo et al. 29
3.2-9 TRNG proposé par Bucci et al. 30
3.2-10 TRNG proposé par Golić. 30
3.2-11 TRNG proposé par Dichtl et al. 32
3.3 TRNG basés sur la métastabilité d'un signal 33
3.3-1 TRNG proposé par Epstein et al. 33
3.3-2 TRNG proposé par Danger et al. 34
3.3-3 TRNG proposé par Vasyltsov et al. 35
3.4 Extracteur d'aléas et décorrélateur 36
4 Evaluation de la qualité d'un générateur vraiment aléatoire 39
4.1 Méthode d'évaluation d'un générateur pseudo-aléatoire 39
4.2 Evaluation d'un générateur vraiment aléatoire 40
5 Conclusions 41

II Evaluation de la qualité de générateurs vraiment aléatoires 43

1 Générateurs vraiment aléatoires sélectionnés 43
2 Evaluation des générateurs au sein d'un FPGA Altera Stratix II 47
2.1 Objectif de l'évaluation 47
2.2 Description et implantation de la batterie FIPS 140-2 49
2.2-1 *Frequency* test 49
2.2-2 *Poker* test 49
2.2-3 *Run* test 50
2.2-4 Long run test 50
2.2-5 Résultats d'implantation pour technologies ASIC et FPGA 51
2.2-6 Résultats de l'évaluation des TRNG par la batterie FIPS 140-2 51
2.2-7 Conclusions de l'évaluation par les tests FIPS 140-2 52
3 Variation de la gigue en fonction du FPGA utilisé et de son activité 55

3.1 Mesure de la gigue cycle à cycle . . . 56
3.1-1 Analyse de la gigue du RO utilisé dans des conditions idéales de fonctionnement . . . 56
3.1-2 Analyse de la gigue du RO soumis à une forte activité parasite 57
3.1-3 Discussion . . . 58
4 Recherche du débit optimal des TRNG en fonction du FPGA ciblé . . . 59
4.1 Objectif . . . 59
4.2 Méthodologie proposée . . . 59
4.3 Implantation matérielle de l'AIS 31 . . . 60
4.3-1 Test T0 : test de différence . . . 61
4.3-2 Test T1 : *frequency* test . . . 63
4.3-3 Test T2 : *poker* test . . . 63
4.3-4 Test T3 : *run* test . . . 63
4.3-5 Test T4 : *long run* test . . . 63
4.3-6 Test T5 : test d'autocorrélation . . . 64
4.3-7 Test T6 : test d'uniformité . . . 64
4.3-8 Test T7 : Evaluation de la distribution des sous-séquence binaire . . . 65
4.3-9 Test T8 : test d'entropie . . . 66
4.3-10 Performance et coût matériel des tests de l'AIS 31 . . . 75
4.4 Evaluation des générateurs sélectionnés . . . 75
4.5 Conclusions de l'évaluation . . . 77
5 Evaluation du coût matériel et de la puissance consommée des TRNG testés 80
5.1 Coût matériel . . . 80
5.2 Puissance consommée . . . 80
6 Conclusions . . . 81

III Vers des RNG uniformes et gaussiens à très haut débit 83

1 Implantation d'un RNG hybride au sein d'un circuit reconfigurable . . . 84
1.1 Description du générateur hybride . . . 84
2 La quête d'un RNG hybride optimal au sein d'un circuit VLSI . . . 85
2.1 *On-CHip Randomness Evaluation* version 1 . . . 85

2.1-1 Architecture du circuit . . . 85
2.1-2 Surface et consommation du circuit . . . 86
2.1-3 Routage du circuit sous l'outil Virtuoso . . . 87
2.1-4 Validation du circuit ASIC . . . 87
2.1-5 Analyse de la source de bruit du générateur . . . 88
2.1-6 Vérification du bon fonctionnement du TRNG . . . 90
2.1-7 Analyse de la qualité du TRNG . . . 91
2.1-8 Conclusions et perspectives . . . 92
2.2 *On-CHip Randomness Evaluation* version 2 . . . 92
2.2-1 Architecture du circuit . . . 92
2.2-2 Performances . . . 93
2.2-3 Discussions et perspectives . . . 94
3 Vers un RNG gaussien et de Rayleigh à très haut débit . . . 95
3.1 Techniques de génération . . . 97
3.1-1 Variables aléatoires gaussiennes . . . 97
3.1-2 Méthode par transformation . . . 98
3.1-3 Sélection d'un algorithme permettant de générer des nombres aléatoires gaussiens à très haut débit . . . 104
3.1-4 Présentation du GRNG proposé par Alimohammad et al. . . . 105
3.2 Implantation de l'algorithme Box-Muller à partir d'une approximation polynomiale minimax . . . 107
3.2-1 Implantation de la fonction $f(u_0)$ . . . 107
3.2-2 Implantation de la fonction $g_0(u_1)$ . . . 115
3.2-3 Résultats d'implantation des fonctions $f(u_0)$ et $g_0(u_1)$ . . . 117
3.2-4 Résultats d'implantation de l'algorithme Box-Muller . . . 117
3.2-5 Validation de la qualité de l'algorithme Box-Muller implanté 118
3.3 Conclusions . . . 120
4 Conclusion . . . 120

IV Recherche d'un générateur pseudo-aléatoire optimisé selon un compromis qualité/coût matériel 123

1 Intérêt des Automates Cellulaires dans les architectures FPGA . . . 123
2 Recherche de la règle optimale d'un CA à 5 voisins . . . 128

2.1 Contexte et objectif . . . 128
2.2 Méthodologie . . . 128
2.2-1 Génération des règles possibles . . . 128
2.2-2 Calcul de l'entropie de l'automate cellulaire . . . 129
2.3 Architecture de la recherche exhaustive . . . 131
2.4 Comparaison de la règle de l'automate cellulaire trouvée avec des générateurs souvent utilisés en VLSI . . . 131
2.4-1 Entropie . . . 132
2.4-2 Distribution des bits aléatoires . . . 133
2.4-3 Conclusion . . . 136
3 Evaluation du coût matériel engendré par l'implantation de tests statistiques très performants . . . 137
3.1 *Birthday test* . . . 138
3.1-1 Présentation . . . 138
3.1-2 Implantation matérielle . . . 139
3.1-3 Calcul du test du χ^2 . . . 140
3.2 *Gorilla test* . . . 141
3.2-1 Présentation . . . 141
3.2-2 *Overlapping 20-tuples bitstream test* . . . 142
3.2-3 Implantation matérielle . . . 142
3.3 *GCD test* . . . 142
3.3-1 Présentation . . . 142
3.3-2 Implantation matérielle . . . 143
4 Conclusions . . . 144

Conclusions et perspectives **147**

Bibliographie **153**

Glossaire **181**

1 Introduction

Contexte

L'objectif d'un générateur de nombres aléatoires (RNG, *Random Number Generator*) est de produire une séquence binaire aléatoire indépendante, imprédictible et uniformément répartie. Les RNG sont nécessaires dans toutes sortes d'applications, par exemple, en cryptographie, télécommunications, simulations numériques, test de circuits VLSI (*Very-Large-Scale Integration*) ou encore dans des algorithmes probabilistes. Le nombre d'applications embarquées nécessitant des RNG matériels augmente constamment, spécialement au sein des circuits reconfigurables, *Field Programmable Gate Array* (FPGA), systèmes sur puce (*System On-Chip* (SoC)) . L'augmentation de la fréquence de ces circuits et la baisse régulière de leur coût ouvrent de nouvelles voies dans le domaine de la conception de générateurs de nombres aléatoires au sein des circuits VLSI.

En vue de la conception matérielle d'un RNG, la solution est d'utiliser un générateur hybride composé d'un générateur vraiment aléatoire (TRNG, *True Random Number Generator*) et d'un générateur pseudo-aléatoire (PRNG, *Pseudo Random Number Generator*). Les PRNG étant basés sur des algorithmes déterministes. Ils sont périodiques et doivent être initialisés par un TRNG. Le vecteur d'initialisation est appelé un germe ou semence (*seed*). Malgré leur périodicité, les PRNG permettent d'atteindre des débits élevés. De plus, en fonction de la qualité du germe, un PRNG est capable de générer une séquence fortement aléatoire. Les TRNG sont quant à eux basés sur une source de bruit physique (e.g. bruit thermique, gigue de phase d'un oscillateur) et dépendent fortement de la qualité de leur implantation. La séquence produite par un TRNG est généralement biaisée, c'est à dire que certaines séquences de nombres aléatoires sont plus fréquemment observées que d'autres. Par conséquent, un processus de post-traitement est souvent ajouté afin de décorréler la séquence produite, engendrant une diminution du débit du générateur ainsi qu'un

sur-coût en termes de surface et de consommation d'énergie du circuit. La majorité des TRNG implantés au sein des circuits VLSI utilisent la gigue produite par un oscillateur ou la métastabilité d'un signal.

Prouver mathématiquement qu'un PRNG est aléatoire est généralement difficile voire impossible. Les générateurs pseudo-aléatoires sont en pratique souvent évalués via l'utilisation de tests statistiques. Plusieurs batteries de tests existent dans la littérature. Les plus connues sont Diehard [MZ93], NIST [RSNMS01] ainsi que les tests statistiques décrits par D. Knuth dans [Knu97]. L'évaluation d'un TRNG est beaucoup plus complexe que celle d'un PRNG. La qualité d'un TRNG est fonction de la source de bruit utilisée, de l'environnement extérieur au circuit et de la qualité d'implantation au sein du circuit ciblé. La méthode la plus élégante lors de la validation d'un TRNG est de fournir un modèle stochastique de la source de bruit utilisée par le générateur [Sch03] [KS08]. L'objectif des modèles stochastiques est de déterminer l'entropie de la source aléatoire ou au moins une borne minimale de celle-ci. Cependant, la complexité mathématique des modèles rend leur utilisation souvent délicate. Par conséquent, deux batteries de tests statistiques sont très généralement utilisées, le FIPS 140-2 [Nat99b] et l'AIS 31 [KS01].

Problématique et contribution

L'objectif de cette thèse est de proposer un RNG possédant un débit élevé, facilement implantable au sein d'un circuit reconfigurable et très robuste. La robustesse du générateur correspond à la qualité constante du TRNG malgré des variations de l'environnement, des attaques externes potentielles et du vieillissement des composants.

Choisir un TRNG parmi tous ceux existants dans la littérature peut s'avérer une tâche délicate. Lors de la validation, il est très rare qu'un concepteur d'un TRNG ne compare les performances du générateur proposé avec celles obtenues par des solutions déjà existantes. La dépendance d'un générateur avec le circuit ciblé, l'implantation effectuée par les concepteurs et l'environnement extérieur dans lequel le circuit a été validé rendent les comparaisons difficiles. A notre connaissance, aucune étude parue dans la littérature ne porte sur la comparaison de la qualité des générateurs existants. Il est par conséquent difficile de sélectionner un TRNG. De plus, dans un contexte matériel (e.g. implantation FPGA), la surface et la consommation du circuit sont des critères importants nécessitant d'être évalués.

Une des problématiques de l'étude est consacrée à l'évaluation de TRNG récemment parus dans la littérature. Les générateurs testés sont sélectionnés pour la possibilité d'implanter ceux-ci au sein d'un maximums de cibles reconfigurables. Une méthode d'évaluation rapide et efficace basée sur l'utilisation de tests statistiques matériels est proposée. Les TRNG, implantés au sein du même circuit reconfigurable (i.e. FPGA) sont analysés en temps réel dans les mêmes conditions réelles d'utilisation (environnement extérieur, bruit d'alimentation). La méthode permet de comparer la qualité des TRNG entres eux. De

plus, la méthode présentée permet d'optimiser les performances des TRNG. À partir des tests statistiques, les débits maximum des TRNG implantés au sein du circuit ciblé sont recherchés. Les débits trouvés, supérieurs à ceux présentés par les concepteurs, montrent l'intérêt d'utiliser les tests statistiques matériels pour pré-caractériser le fonctionnement d'un TRNG et optimiser leur débit.

À partir de l'étude précédente, un TRNG est sélectionné pour ses propriétés statistiques. Une architecture de générateur hybride permettant de produire des nombres aléatoires uniformes à très haut débit est ensuite proposée. Cependant, comme le montre le chapitre II, malgré la qualité du générateur sélectionné, l'aléa en sortie du TRNG peut varier au cours du temps à cause d'attaques externes possibles sur le circuit, de la variation des conditions extérieures ou encore du vieillissement des composants. La qualité aléatoire de la séquence de germes produite par un TRNG est primordiale pour assurer l'imprédictibilité d'un PRNG. Afin d'améliorer la robustesse d'un TRNG, des mécanismes de tests statistiques en ligne doivent être mis en œuvre. Ainsi, une évaluation en temps réel du comportement du générateur est réalisée et les séquences défaillances générées par celui-ci sont évitées. Cette étude propose d'utiliser une implantation efficace de la batterie du FIPS 140-2. La surface négligeable du circuit au sein des FPGA actuels et les performances obtenues par le composant permettent de contrôler la qualité du TRNG sélectionné en temps réel.
Après sélection d'un TRNG pour des performances et de mécanismes de test en ligne, un PRNG de très bonne qualité est sélectionné. Le principe du générateur hybride produisant des nombres aléatoires uniformes à très haut débit est présenté dans la figure 0-1. Une implantation du générateur au sein d'un circuit reconfigurable et au sein d'un circuit ASIC est réalisée.

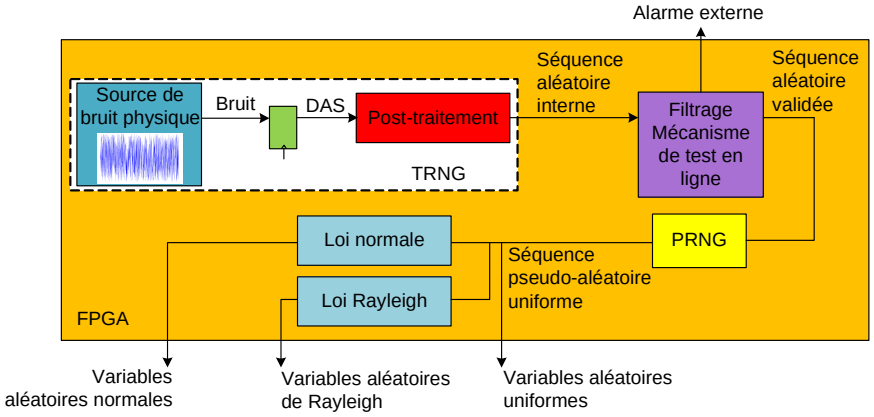

Figure 0-1 – Présentation du générateur proposé. Le circuit, implanté au sein d'un FPGA, est composé d'un TRNG et d'un PRNG. La séquence aléatoire produite par le TRNG est analysée en temps réel, par des mécanismes de test en ligne. En cas de défaillances importantes du générateur, une alarme est levée et les séquences déficientes sont supprimées. L'analyse permet d'assurer un vecteur d'initialisation très aléatoire.

L'étude menée sur le RNG hybride a débouché sur un partenariat industriel réalisé avec une entreprise spécialisée en cryptographie à travers le projet pôle images et réseaux SPRING (*Shelf Proof Random Integrated Number Generator*). L'entreprise propose une solution permettant d'assurer la confidentialité, l'intégrité et/ou la non interception des données transmises à travers un canal de transmission. La solution est basée sur l'algorithme des masques jetables. Le principe de fonctionnement est de chiffrer chaque texte clair à partir d'une clé aléatoire dont la longueur est supérieure ou égale à celle du message. Chaque clé n'est utilisée qu'une seule fois. Le nombre de clés requis, la longueur de celles-ci et la transmission des clés à travers le canal de transmission engendrent une mise en œuvre très coûteuse. L'utilisation d'une clé unique par chiffrement impose un débit élevé au générateur de nombres aléatoires et une qualité irréprochable.
Dans le cadre du projet SPRING, notre objectif a été de proposer à l'entreprise de cryptographie un RNG facilement implantable au sein d'une multitude de circuits reconfigurables et des mécanismes de contrôle de qualité peu coûteux en termes de surface. Le circuit développé est actuellement utilisé au sein de la solution proposée par l'entreprise. De plus, une clé usb générant des nombres aléatoires uniformes sera commercialisée sous peu par l'entreprise à partir des travaux réalisés.

Le circuit présenté dans la figure 0-1 permet de générer des nombres aléatoires uniformes à haut débit. Dans certaines applications, il est parfois nécessaire de générer des nombres aléatoires suivant une distribution de probabilité différente. Par exemple, comme le présente la figure 0-2, en communications numériques, il est souvent nécessaire de connaître le taux d'erreur binaire d'une chaîne de transmission. Le taux d'erreur binaire correspond à une mesure relative du nombre d'erreur binaire mesuré en sortie d'une transmission numérique. Les erreurs sont généralement engendrées par le canal de transmission [Pro00] et des modèles de canaux de transmission sont souvent utilisés pour simuler le comportement de tel système. Cependant, les taux d'erreurs binaires actuels (i.e. 10^{-9}) entraînent la réalisation d'un grand nombre de simulations (i.e. supérieur à 10^{14}). La complexité des modèles de canaux de transmission engendre des temps de calcul très importants. Une alternative pour améliorer la rapidité des simulations est d'implanter le modèle de canal de transmission en matériel, au sein de circuits reconfigurables. En communications numériques, un modèle de canal de transmission souvent utilisé est un canal avec bruit gaussien additif. Celui-ci nécessite l'utilisation d'un générateur de bruit gaussien (GRNG). L'implantation matérielle d'un GRNG est délicate car des fonctions complexes sont souvent utilisées par les algorithmes. De plus, la précision élevée requise peut engendrer une surface non négligeable. À partir de l'utilisation du circuit présenté dans la figure 0-1, l'implantation d'un générateur de nombres aléatoires gaussiens à très haut débit est présenté. Par ailleurs, le circuit implanté permet de générer efficacement des nombres aléatoires suivant une loi de Rayleigh.

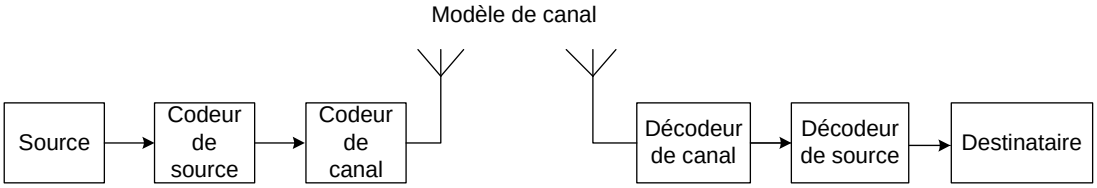

Figure 0-2 – Système de communications numériques avec processus de compression des données et de protection contre les erreurs de transmission.

Plan du mémoire

Ce travail de recherche est présenté au travers de quatre chapitres. Le premier chapitre introduit le concept de générateurs de nombres aléatoires. Le principe de fonctionnement et les applications des générateurs sont présentés. Ceci permet de comprendre l'importance de générer une séquence fortement aléatoire. Un état de l'art des générateurs vraiment aléatoires est réalisé. Cette partie présente les limites des TRNG actuels. Enfin, les méthodes d'évaluation des TRNG sont explicitées.

Le deuxième chapitre est consacré à l'évaluation sur cible FPGA de TRNG récemment parus dans la littérature. Ce chapitre étudie la variation de la qualité aléatoire des TRNG analysés en fonction de la cible visée, de l'activité du circuit et de l'environnement extérieur. De plus, la surface et la consommation des TRNG sont évaluées. Ce chapitre permet la sélection d'un TRNG d'excellente qualité.

Le troisième chapitre présente l'implantation du générateur hybride proposé. Le RNG est composé d'un TRNG sélectionné pour ses propriétés statistiques, de mécanismes de test en ligne et d'un PRNG réputé. Après analyse des performances obtenues par le système, un simulateur matériel d'un canal à bruit gaussien additif et d'un canal de Rayleigh est réalisé sur cible FPGA et ASIC.

Le quatrième chapitre décrit une étude annexe réalisée au cours de la thèse consistant à rechercher un générateur pseudo-aléatoire optimisé selon un compromis qualité/coût matériel. L'objectif est de trouver un générateur pseudo-aléatoire profitant de la structure "systolique" d'un FPGA. Les automates cellulaires possèdent une structure intéressante pour atteindre des débits maximum. Cependant, malgré leur simplicité, prédire le comportement d'un CA en fonction de la règle utilisée est très difficile. Ce chapitre propose une solution permettant la recherche exhaustive de la règle optimale d'un automate cellulaire à 5 voisins.

Enfin, le dernier chapitre conclut ce document en résumant les apports des travaux réalisés et les perspectives qui s'en dégagent.

Chapitre

1 État de l'art

1 Introduction

Cette partie introduit le concept de générateur de nombres aléatoires. Dans un premier temps, après avoir défini la notion de séquence aléatoire, quelques applications nécessitant l'utilisation d'un RNG sont présentées. Dans la section 3, un état de l'art des TRNG implantés au sein des circuits reconfigurables (FPGA) est effectué. Cette partie permet de comprendre les limites des TRNG actuels. Enfin, la section 4 réalise une synthèse des méthodes de validation d'un TRNG.

2 Rôle et principe de fonctionnement

2.1 Définitions préliminaires

Séquence binaire aléatoire

Une séquence binaire x constituée de l bits est dite aléatoire lorsque les l bits sont indépendants, imprédictibles et équiprobables [Knu97].

Différence entre une séquence de nombres aléatoires et une séquence de bits aléatoires

Par abus de langage, un générateur de nombres aléatoires et un générateur de bits aléatoires désignent souvent la même notion. Leur objectif est de générer une séquence de mots aléatoires. Les mots appartiennent à un alphabet fini $\mathcal{A}$. Dans le cas d'un générateur de bits aléatoires, l'alphabet est de dimension 2 et $\mathcal{A} = \{0, 1\}$. Afin d'obtenir des nombres aléatoires, il est possible de regrouper les bits successifs en sortie du générateur de bits aléatoires.

Générateur de nombres aléatoires (RNG)

Un générateur de nombres aléatoires est un algorithme déterministe ou indéterministe permettant de générer une séquence binaire x uniforme, imprédictible et indépendante. Prouver mathématiquement qu'un générateur est aléatoire est souvent difficile. Par conséquent, des batteries de tests statistiques sont utilisées afin de vérifier si la séquence générée est réellement aléatoire.

Deux grandes familles de générateurs existent : les générateurs vraiment aléatoires (TRNG) et les générateurs pseudo-aléatoires (PRNG). Les TRNG sont basés sur des sources de bruit physique non déterministes. Les sources de bruit généralement utilisées en électronique sont le bruit thermique, la gigue produite par un oscillateur ou la métastabilité d'une bascule. La gigue correspond à la déviation d'un signal par rapport à son comportement idéal. La séquence générée par un TRNG est généralement biaisée. Un étage de post-traitement est souvent ajouté afin de réduire la corrélation. L'ajout de ce composant engendre une diminution du débit du TRNG ainsi qu'une augmentation de la puissance consommée.

Les PRNG sont des algorithmes déterministes basés sur une équation. Un grand nombre de PRNG existent dans la littérature. On note en particulier les générateurs congruentiels linéaires, le générateur Mersenne Twister, Fibonacci, Blum Blum Shub et bien d'autres. Un état de l'art des PRNG existants est réalisé dans [Knu97]. Les PRNG produisent une séquence périodique. Malgré leur périodicité, les PRNG sont souvent utilisés en VLSI car ils permettent d'atteindre des débits très élevés. Cependant, les PRNG sont initialisés par les TRNG. Le vecteur d'initialisation est appelé une semence ou germe (*seed*). En fonction de la qualité de celle-ci, un PRNG est capable de produire une séquence ayant une entropie élevée.

La figure 1-1 présente un RNG hybride composé d'un TRNG et d'un PRNG.

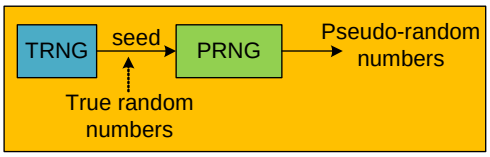

Figure 1-1 – Générateur aléatoire hybride composé d'un TRNG et d'un PRNG.

Par convention, au sein de ce document, les variables aléatoires sont notées en lettres capitales et les réalisations de celles-ci, en lettres minuscules.

2.2 Applications

Dans cette section, trois domaines nécessitant l'utilisation de générateurs de nombres aléatoires sont présentés : la cryptographie, les simulations de Monte-Carlo et les commu-

nications numériques. L'objectif n'est pas d'expliquer les subtilités de ces applications mais plutôt de présenter l'importance des générateurs de nombres aléatoires.

2.2-1 Cryptographie

Généralités

La cryptographie est une discipline de la cryptologie. L'objectif est de sécuriser la transmission d'informations, son stockage ou encore son authentification. La confidentialité, l'intégrité des données, l'authentification d'une entité ou encore la connaissance de l'origine d'une donnée sont des tâches qu'un système cryptographique doit réaliser avec succès. Utilisée depuis l'antiquité, la cryptographie connaît un engouement très important depuis les années 70. Ceci a été favorisé par les travaux de Shannon en 1949 [Sha49], qui ont permis de poser les bases théoriques de la cryptographie moderne.

Pour présenter les algorithmes de cryptographie, cette partie repose sur une communication entre deux interlocuteurs Alice et Bob. Le système cryptographique est dit sûr, si aucun ennemi (Oscar dans l'exemple) n'est capable d'accéder aux informations secrètes échangées entre Alice et Bob. Alice souhaite émettre un message x à Bob. x doit être chiffré à partir d'une clé notée e. Le chiffrement est rendu possible grâce à une fonction de chiffrement notée E_e. Le message chiffré y est obtenu par

$$y = E_e(x). \tag{1-1}$$

A la réception du message y, Bob doit le déchiffrer. Le déchiffrement est réalisé par une fonction de déchiffrement, D_d. Le message x est obtenu par

$$x = D_d(y). \tag{1-2}$$

Un système cryptographique est défini par :

- un ensemble fini $\mathcal{M}$ de blocs de textes clairs ;
- un ensemble fini $\mathcal{C}$ de blocs de textes chiffrés ;
- un ensemble fini $\mathcal{K}$ de clés ;
- un ensemble de fonctions de chiffrement $E_e : \mathcal{M} \to \mathcal{C}$ tel que E_e soit injective [1] ;
- un ensemble de fonctions de déchiffrement $D_d : \mathcal{C} \to \mathcal{M}$ tel que $E_e \circ Dd = \mathbb{ID}$, où $\mathbb{ID}$ représente la fonction identité.

A partir de l'expression de l'entropie d'une variable aléatoire X

$$H(X) = -\sum_{i=1}^{n} p_i \log p_i, \tag{1-3}$$

Shannon définit la notion de sécurité inconditionnelle. La variable aléatoire X est discrète et peut prendre n valeurs possibles : $x_1, ..., x_n$ avec une probabilité non nulle donnée par

$$p_i = P(X = x_i). \tag{1-4}$$

1. Une application $f : E \to F$ est injective si $\forall (x, y) \in E^2$, $\forall x \neq y$, $f(x) \neq f(y)$.

L'entropie $H(X)$ est une mesure de l'incertitude. L'entropie est maximale lorsque la variable aléatoire X est équidistribuée sur $\{x_1, ..., x_n\}$. Par opposition, $H(X)$ est minimale si X est presque sûrement constante.

Shannon considère les messages de textes clairs possibles $x \in \mathcal{M}$ comme étant les réalisations d'une variable aléatoire X et les textes chiffrés $y \in \mathcal{C}$ comme les réalisations d'une variable aléatoire Y. Oscar est susceptible de connaître les ensembles $\mathcal{M}, \mathcal{C}, \mathcal{K}$ ainsi que le texte chiffré y. Par conséquent, la sécurité du système ne peut pas reposer seulement sur les fonctions de chiffrement et de déchiffrement. Shannon démontre qu'un système est dit inconditionnellement sûr lorsque $H(X|Y) = H(X)$. La connaissance d'un message chiffré n'apporte aucune information sur le message clair. De plus, si K représente la variable aléatoire pouvant prendre les différentes valeurs de la clé $k \in \mathcal{K}$, il est possible de démonter que $H(X|Y) \leq H(K|Y)$. Cette expression signifie que l'incertitude sur X connaissant Y est inférieure ou égale à l'incertitude sur K connaissant Y. De plus, dans un système inconditionnellement sûr, $H(X|Y) = H(X)$ donc $H(K) \geq H(X)$. Ce résultat implique que la sécurité d'un système inconditionnellement sûr repose sur la qualité aléatoire de la clé. D'après [KIKS04], la résistance d'un algorithme cryptographique face à un ennemi est fonction du choix et de la politique de gestion des clés aléatoires. La définition d'un système inconditionnellement sûr peut aussi être vue de la manière suivante. Soit deux messages quelconques appartenant à $\mathcal{M}$: x_1 et x_2, la probabilité que $y \in \mathcal{C}$ soit le chiffrement de x_1 doit être égale à la probabilité que y soit le chiffrement de x_2. D'après [Cor06], cette sécurité n'est possible que si la longueur en nombre de bits du message x est inférieure ou égale à celle de la clé aléatoire. Ceci est démontré à partir de l'inéquation $H(K) \geq H(X)$. D'après [Cor06], cet inconvénient constitue une limitation importante. Effectivement, l'idéal serait de chiffrer un message à l'aide d'une clé de longueur faible. D'après [Cor06], la définition d'un système inconditionnellement sûr exprimée par Shannon ne peut être utilisée. La sécurité d'un système doit être en relation avec les moyens informatiques dont dispose un ennemi.

Deux grandes familles d'algorithmes cryptographiques existent : la cryptographie symétrique et la cryptographie asymétrique [KIKS04].

Cryptographie symétrique

Les algorithmes de cryptographie symétrique utilisent des clés privées. Soit un ensemble fini de fonctions de chiffrement $\{E_e : e \in \mathcal{K}\}$ et un ensemble fini de fonctions de déchiffrement $\{D_d : d \in \mathcal{K}\}$. L'algorithme de chiffrement est dit symétrique si à partir de la clé e, il est possible de déterminer facilement la clé d. Généralement, les algorithmes symétriques utilisent les mêmes clés lors du chiffrement et du déchiffrement, soit $e = d$. Dans un algorithme à clé privée (ou symétrique), la sécurité repose sur la clé. Alice et Bob possèdent tous les deux la même clé pour chiffrer et déchiffrer un message. La clé doit rester inconnue des ennemis. Par conséquent, la clé e utilisée par Alice doit être envoyée à Bob grâce à un canal sécurisé. Ainsi, la clé doit être aléatoire et être renouvelée lors de chaque changement

de session ou régulièrement durant une longue session.

Le chiffrage de l'information est généralement réalisé par des chiffrement par bloc. Un chiffrement par bloc E est une fonction prenant en paramètre une clé e et un texte clair x d'une longueur finie. E génère un texte chiffré de longueur finie [Cor06] à partir d'un message en clair et d'une clé (ou chiffrement en bloc). Deux types de chiffrement par bloc existent, les chiffrements par substitution et les chiffrements par transposition. Un chiffrement par substitution est un chiffrement par bloc dans lequel les symboles sont remplacés par d'autres symboles. Un chiffrements par transposition est un chiffrement par bloc dans lequel est réalisée une permutation sur les symboles de chaque bloc à chiffrer. [MVO96] explique en détail le fonctionnement des chiffrement par bloc. Un autre moyen de chiffrer une information dans un algorithme à clé privée est d'utiliser des chiffrements par flot. Les chiffrements par flot sont des chiffrements par bloc de longueur égale à un. Si la probabilité d'erreur de transmission sur un canal est élevée, les chiffrements par flot permettent de ne pas propager d'erreur de transmission. Lors de l'utilisation d'un algorithme à clé privée, il a été vu que le problème est de réaliser l'échange de la clé entre Alice et Bob. La nécessité d'avoir un canal sécurisé est une contrainte forte. Pour éviter cela, il est possible d'utiliser un protocole de Diffie-Hellman [MVO96].

Les algorithmes DES (*Data Encryption Standard*) [Nat99a], 3DES (*Triple DES*) [Nat08] et AES (*Advanced Encryption Standard*) [Nat01] sont des exemples de systèmes à cryptographie symétrique.

Cryptographie asymétrique

Les algorithmes de cryptographie asymétrique diffèrent entre autres dans la gestion de la clé. Soit un ensemble fini de fonctions de chiffrement $\{E_e : e \in \mathcal{K}\}$ et de déchiffrement $\{D_d : d \in \mathcal{K}\}$. La technique repose sur l'hypothèse suivante. Quel que soit le message émis par Alice $y \in \mathcal{C}$, pour toutes les paires de transformation (E_e, D_d), il est impossible (ou très difficile) de trouver le message clair $x \in \mathcal{M}$ tel que $E_e(x) = y$. Par conséquent, pour tout e, il est impossible de trouver d. Si Alice et Bob désirent communiquer ensemble, Bob sélectionne une paire de clés (e, d) et envoie e à Alice par un canal non sécurisé. Alice, après avoir reçu e, envoie un message x à Bob. Alice calcule $y = E_e(x)$ puis le fait parvenir à Bob. A la réception, Bob calcule la transformation inverse $x = D_d(y)$. La sécurité repose sur le fait que, même connaissant e, un attaquant n'est pas capable de calculer d.

La cryptographie asymétrique (ou clé publique) nécessite d'authentifier les émetteurs des messages. Afin de montrer la nécessité d'introduire un protocole d'authentification, un exemple est donné. Un échange d'information est réalisé entre Alice et Bob. Oscar va remplacer Alice aux yeux de Bob. Oscar envoie à Alice une clé notée e_1. Alice envoie un message à Bob. Oscar intercepte le message émis puis le déchiffre avec sa propre clé d_1. Étant donné qu'Oscar a intercepté la clé de Bob e, il est capable de chiffrer le message venant d'Alice. Par conséquent, Oscar envoie le message à Bob. Ce scénario montre l'intérêt de mettre en place une authentification des clés publiques. Si Alice désire émettre un

message à Bob, elle doit être sûre de chiffrer son information par la clé de Bob. La procédure d'authentification est détaillée dans [MVO96].

Les algorithmes RSA (issu du nom des inventeurs Rivest, Shamir et Adleman) [RSA78] et ECC (*Elliptic Curve Cryptosystems*) [HMV03] sont des exemples de système à cryptographie asymétrique. Un exemple d'implémentation d'un algorithme à clef asymétrique dans la sécurisation des réseaux GSM, GPRS et UMTS est donné dans [GMV03].

Attaques d'un système cryptographique

Lors de l'implantation d'algorithmes cryptographiques en matériel ou en logiciel lorsque ceux-ci sont exécutés sur un processeur, par exemple, ceux-ci présentent parfois des aléas de fonctionnement [MCL06]. Les systèmes sont par conséquent vulnérables aux attaques d'ennemis. Les deux grandes familles d'attaques sont les attaques invasives (*invasive attacks*) (IA) et les attaques non invasives (*non-invasive attacks*) (NIA).

Les IA réalisent du *reverse engineering* et nécessitent du matériel de laboratoire conséquent. Les NIA profitent des faiblesses de l'implantation matérielle, comme par exemple les variations du temps d'exécution, de la puissance consommée ou encore des émissions électromagnétiques [MCL06]. A partir du/des relevés de ces différentes informations issues de l'exécution de l'algorithme, l'extraction de paramètres statistiques peut révéler certaines inconnues du système. Deux méthodes sont reconnues dans la littérature : la *differential power analysis* (DPA) [KJJ99] et la *differential fault analysis* (DFA). La DPA exploite la puissance consommée et la DFA tente d'injecter des fautes dans le système pour trouver ses caractéristiques [BECH$^+$06]. Par exemple, les *glitch attacks* tentent d'influencer l'alimentation, la masse ou l'horloge afin de déstabiliser le système. Une autre injection de fautes est réalisée par des attaques à lumières plus ou moins visibles qui entraînent des perturbations au niveau des semi-conducteurs. D'après [MCL06], certaines solutions permettent de lutter contre les attaques par DPA en diminuant la puissance consommée ou en modifiant l'algorithme du cryptosystème afin de décorréler les états des entrées et la puissance consommée. Par contre, d'après [SPRQ06], lutter contre ces attaques est difficile. Diminuer la puissance n'exclut pas totalement le risque d'une attaque par DPA ou bien sûr par d'autres méthodes : les *timing attacks*, les DFA ou encore les attaques par lumière.

De plus, l'implantation des algorithmes cryptographiques, comme l'AES, le DES ou encore le RSA au sein des cartes à puces (*smart-cards*) représente une cible facile pour les DFA. Ces circuits sont optimisés afin de rendre leur intégration élevée tout en minimisant leur consommation. D'après [MCL06], ces circuits présentent par conséquent une faiblesse à l'injection de fautes. Après avoir montré le principe de fonctionnement d'une recherche de clé dans le DES, [MCL06] présente une architecture tentant de protéger l'algorithme contre les DFA. Un co-processeur est rajouté afin de détecter et de corriger les fautes de transitions. Le co-processeur réplique les parties vulnérables du système aux attaques par fautes. [BECH$^+$06] présente les méthodes utilisées pour réaliser une injection de fautes afin de casser la sécurité d'algorithmes cryptographiques reconnus, comme le RSA, le DES ou

encore le protocole d'échange de clé dans le DES. Des solutions matérielles et logicielles permettant de lutter contre l'injection de fautes sont présentées.

[SPRQ06] présente une autre approche que la DPA permettant d'attaquer un système à partir de sa puissance consommée. Le principe de fonctionnement est d'effectuer une analyse statistique basée sur la réalisation de mesures de corrélation. La technique, nommée *correlation power analysis* (CPA) permet une meilleure utilisation du courant de fuite. [SPRQ06] présente une analyse de la CPA d'une manière théorique. La puissance consommée en CMOS étant proportionnelle à la fréquence des transitions, l'objectif est de prédire le nombre de transitions qu'engendre une clé. La corrélation entre la prédiction et la puissance mesurée est calculée. Si la corrélation est maximale, la clé est trouvée. La méthode possède l'inconvénient de réaliser beaucoup de calculs d'autocorrélation. [SPRQ06] présente quatre méthodes permettant de lutter contrer les attaques utilisant la puissance consommée. Deux de celles-ci nécessitent un générateur de nombres aléatoires. Par contre, aucune méthode ne permet de prouver la sécurité théorique du système [SPRQ06]. La seule solution permettant d'atteindre un niveau de sécurité acceptable est de réaliser une combinaison de plusieurs techniques.

Beaucoup de méthodes d'attaque de systèmes cryptographiques existent dans la littérature. L'objectif ici est de prendre conscience de la vulnérabilité des systèmes cryptographique face aux attaques.

Cette section a présenté des techniques utilisées en cryptographie. L'objectif n'était pas d'expliquer les méthodes existantes mais plutôt de présenter la problématique de cette discipline. Il a été vu qu'en pratique, la sécurité inconditionnelle d'un système est difficilement démontrable. Par conséquent, des méthodes d'attaques permettent de casser les cryptosystèmes. D'après [BMV06], chaque module cryptographique a une durée de vie limitée allant de cinq à dix ans. De plus, cette partie a montré l'importance des clés aléatoires dans les algorithmes cryptographiques. Dans ces applications, la faiblesse d'une séquence aléatoire peut provoquer la défaillance totale de la sécurité du système. Ceci impose d'utiliser des générateurs de nombres aléatoires de qualité.

L'implantation au sein des circuits embarqués de modules cryptographiques est une réelle demande. Malheureusement, l'implantation de protocoles de signature numérique ou d'algorithmes de cryptographie est inadaptée à ces circuits (consommation d'énergie et faible densité d'intégration). Par conséquent, travailler sur l'implantation des algorithmes de cryptographie est nécessaire afin d'adapter le besoin en sécurité de ces circuits avec leurs possibilités [BMV06].

Un grand nombre d'applications cryptographiques reposent sur les propriétés statistiques d'une clé (imprédictibilité). En cas de défaillance de celle-ci, le système peut être totalement vulnérable. Ceci montre l'importance de la génération des clés aléatoires au sein des circuits cryptographiques.

2.2-2 Méthodes de Monte-Carlo

Il est parfois difficile d'analyser un processus physique ou un modèle mathématique. Dans certains cas, l'utilisation de nombres aléatoires est nécessaire pour résoudre un problème difficile [Jam80]. La méthode de Monte-Carlo est une technique de simulation numérique très utile et utilisée pour approcher l'intégrale d'une fonction multi-dimensionnelle. La méthode est basée sur la théorie des grands nombres. Pour approcher

$$\frac{1}{b-a}\int_a^b f(u)du, \tag{1-5}$$

il est possible de tirer au hasard n nombres aléatoires $x_i \in [a,b]$ en suivant une loi uniforme. Pour n tendant vers l'infini, le résultat de

$$\frac{1}{n}\sum_{i=1}^{n} f(x_i) \tag{1-6}$$

converge vers le résultat désiré et résoud l'intégrale (1-5). Les méthodes de Monte-Carlo peuvent être vues comme un problème d'estimation. L'estimateur de Monte-Carlo de (1-5) est donc (1-6). L'objectif est d'avoir un estimateur non biaisé minimisant la variance. La lenteur de la convergence de la méthode est l'inconvénient majeur. Par contre, celle-ci est insensible à la dimension du problème (de la fonction à intégrer) et à la régularité de la fonction. Pour réduire le temps de convergence, il est possible d'utiliser des techniques de diminution de la variance (*variance-reducing techniques*) [Jam80]. L'inconvénient de ces techniques est qu'elles nécessitent des connaissances a priori sur les statistiques du problème. Par conséquent une autre technique est utilisée : la réduction de la variance de manière adaptative (*adaptative variance-reducing techniques*) [Jam80].

Les méthodes de Monte-Carlo nécessitent donc un générateur de nombres aléatoires performant et de qualité. Dans le passé, la faible connaissance de la théorie des générateurs de nombres aléatoires a été un frein aux simulations de Monte-Carlo. La qualité de ceux-ci était relativement faible, leurs périodes étaient petites et la sortie des RNG était souvent corrélée. Par conséquent, une extension de la méthode de Monte-Carlo a été conçue : les méthodes de quasi Monte-Carlo (QMC). Ces méthodes relâchent des contraintes sur la génération des variables aléatoires. L'objectif des générateurs est de remplir des conditions propres aux simulations de QMC. Par exemple, la corrélation présente en sortie de RNG est admise. De plus, la variation de l'uniformité des variables aléatoires (v.a.) pour certaines distributions doit être inférieure à celle d'une v.a. générée par un générateur vraiment aléatoire.

Dans certaines simulations, la v.a. générée doit suivre une autre loi que la loi uniforme. Par exemple, il est souvent nécessaire que la distribution de la v.a. suive une loi : gaussienne, exponentielle, binomiale ou encore de Poisson. [Jam80] et [ST91] expliquent comment obtenir de telles lois à partir d'une série de v.a. uniformes.

2.2-3 Communications numériques

Utilisés lors de la simulation d'un canal de transmission (radio, filaire, satellite), les générateurs de nombres aléatoires permettent d'approcher la réponse fréquentielle du canal de transmission. Par ailleurs, les RNG sont nécessaires lors de l'estimation d'un canal de transmission. Pour obtenir la réponse impulsionnelle d'un canal, une méthode simple est d'attaquer celui-ci par un échelon. En pratique, une séquence aléatoire est préférée à un échelon car celle-ci possède une puissance moyenne supérieure et par conséquent un meilleur rapport signal à bruit (SNR) pour la même puissance transmise [Skl01]. [CLLL06] montre l'importance d'utiliser une séquence pseudo-aléatoire afin de traquer la variation de la réponse impulsionnelle du canal de transmission.

Les RNG sont requis lors d'un étalement de spectre (*spread-spectrum techniques*) utilisant la technique directe [Skl01]. La technique est requise par exemple dans les communications militaires. Afin de dissimuler l'information transmise, celle-ci est codée grâce à une séquence pseudo-aléatoire (*pseudonoise sequence*) [Skl01].

Les générateurs de nombres aléatoires sont aussi utilisés pour réaliser la simulation d'un canal de communication (e.g. canaux MIMO [VY03]). Généralement, la loi utilisée est une loi normale. Une loi normale est obtenue à partir d'une loi uniforme. Plusieurs techniques permettent de réaliser cette transformation. Celles-ci reposent généralement sur l'utilisation d'une fonction non-linéaire. Le principe de base de cette transformation repose sur le théorème central limite. La somme de n variables aléatoires uniformes suit une loi normale lorsque n est grand. La lenteur de la méthode est un frein à son utilisation. Afin d'améliorer la rapidité, d'autres techniques ont été étudiées. La méthode de Wallace [Wal96] possède l'avantage de ne pas nécessiter de fonctions non-linéaires. La méthode exprime la sortie du générateur à l'étape i en fonction des sorties précédentes. Par conséquent, la séquence générée est corrélée [Lee06]. La corrélation peut être diminuée à un niveau acceptable en choisissant les paramètres optimaux à cette méthode [Lee06]. [LLV+05] et [LLVC04] présentent une implantation matérielle de la méthode de Wallace. Une autre méthode est l'acceptation-refus (*rejection-acceptance method*). L'inconvénient de la méthode est qu'elle utilise des boucles conditionnelles. Par conséquent, le débit en sortie n'est pas régulier. La méthode de Box-Muller, avec ou sans utilisation du théorème central limite, est une alternative aux deux solutions précédentes. La méthode consiste à réaliser des transformations non-linéaires sur la séquence uniforme de départ. Les fonctions nécessaires sont le logarithme, la racine carrée, le cosinus et le sinus. [BDG03] et [LLVC04] ont implanté la méthode de Box-Muller au sein d'un FPGA. Afin de corriger les erreurs de calculs lors de l'implantation en virgule fixe, [BDG03] et [LLVC04] utilisent le théorème central limite. [ABS05] propose aussi une implantation matérielle de la méthode. Une extension de la technique de Box-Muller existe, il s'agit de la méthode de Box-Muller modifiée (aussi appelée *polar method*) [ST91]. L'inconvénient de la méthode de Box-Muller modifiée est que celle-ci repose sur une méthode d'acceptation-refus. Par conséquent, le débit en sortie n'est pas régulier. [ST91] présente aussi la méthode de Ziggurat. Enfin la dernière technique est

d'utiliser la fonction de répartition inverse de la loi normale. Une table de correspondance permet de réaliser directement la transformation. La solution est difficilement réalisable en raison du coût mémoire [LLV+05]. Il est parfois nécessaire de générer des variables aléatoires corrélées. [MC94] présente une méthode générant deux variables aléatoires uniformes dont la corrélation est contrôlée. Afin de tester la qualité de la distribution des lois normales obtenues, la séquence binaire est analysée par des tests d'adéquation (*goodness-of-fit test*). D'après [LLV+05] et [LLVC04], les tests du Chi-deux (ou Chi-carré χ^2) et d'Anderson-Darling sont performants pour tester la distribution obtenue.

3 Générateurs de nombres vraiment aléatoires

Comme présenté dans la figure 1-2, un TRNG peut être décomposé en trois parties, le bruit physique, un échantillonneur et un module de post-traitement [KS01]. Les signaux en sortie de l'échantillonneur et du post-traitement sont souvent appelés bruit analogique échantillonné (noté DAS pour *Digitized Analog Signal*) et nombres aléatoires internes. Les réalisations des v.a. DAS et des nombres aléatoires internes sont notées respectivement $r_1, r_2, \ldots$ et $y_1, y_2, \ldots$.

Une unité de post-traitement est souvent requise dans la construction des TRNG. Son rôle est de corriger les imperfections présentes sur le signal DAS. Un grand nombre de techniques existent dans la litérature, tel que les extracteurs de Von Neumann, les fonctions de hachage ou les extracteurs d'aléa.

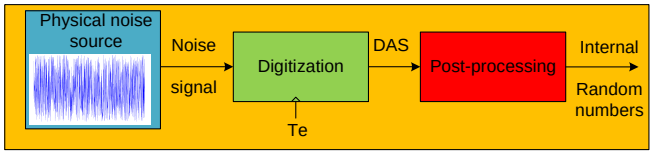

Figure 1-2 – Décomposition d'un générateur vraiment aléatoire. Un TRNG peut être décomposé en trois parties, le bruit physique, un échantillonneur et un module de post-traitement

La plupart des TRNG basés sur des techniques électroniques et implantés sur cible VLSI utilise la gigue de phase (ou *jitter*) d'un ou plusieurs oscillateurs. La gigue désigne la déviation d'un signal par rapport à son comportement idéal. La gigue est causée par des bruits déterministes (bruit d'alimentation, bruit de couplages) et aléatoires (bruit de substrat, variation de la température). Par ailleurs, la gigue dépend fortement de l'oscillateur utilisé au sein du TRNG (souvent une boucle à verrouillage de phase (PLL) ou un oscillateur à anneau) [Fah05]. En pratique, la gigue peut être influencée par des bruits externes au circuit et par son activité. Cette faiblesse rend certaines attaques réalisables, par exemple en soumettant le circuit à des conditions externes qualifiées d'hostiles.

Par rapport aux nombreux PRNG existants, un nombre limité de TRNG implantés sur cible matérielle FPGA et ASIC existent dans la littérature. Au sein des circuits, la technique la plus utilisée est basée sur l'échantillonnage d'oscillateurs (*Oscillator Sampling*) (OS). Cependant, un TRNG basé sur une amplification directe d'un bruit physique ou sur l'utilisation d'un signal métastable est parfois rencontré.

3.1 TRNG basés sur l'amplification d'un bruit physique

3.1-1 TRNG proposé par Bagini et al.

Bagini et al. [BB99] présentent un TRNG utilisant une source de bruit analogique externe et un échantillonneur numérique. Le schéma de principe du générateur est présenté dans la figure 1-3. [BB99] fournit un modèle analytique du générateur exprimant la relation entre le débit de celui-ci et la corrélation de la sortie aléatoire. Il est montré que le TRNG est résistant aux variations des conditions externes et des composants. De plus, d'après [BB99], il n'est pas nécessaire de contrôler le générateur via l'utilisation de tests statistiques. Cependant, aucun résultat d'implantation du TRNG n'est fourni.

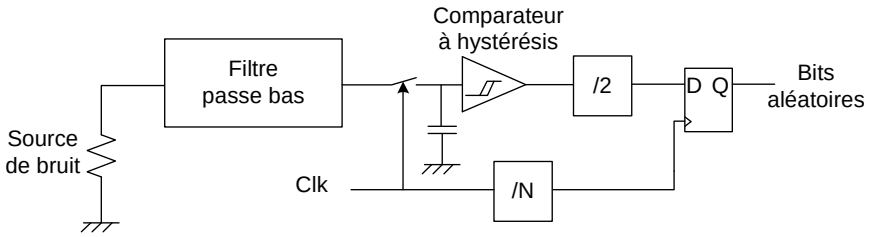

Figure 1-3 – Schéma de principe du TRNG [BB99]. Les principaux blocs composant le TRNG sont une source de bruit physique externe, un étage de filtrage et un comparateur à hystérésis.

3.1-2 TRNG proposé par Jun et al.

Un autre exemple d'amplification directe d'une source de bruit physique a été proposé par Intel dans [JK99]. Le schéma de principe du générateur est présenté dans la figure 1-4. Le bruit thermique du générateur est obtenu en soustrayant les signaux relevés en sortie de deux résistances. Ceci améliore la robustesse du système contre les variations de l'alimentation et de l'environement extérieur. La différence des deux signaux impliquant un bruit thermique est ensuite amplifiée. Le signal obtenu est appliqué en entrée d'un oscillateur contrôlé en tension (VCO). La sortie du VCO est échantillonnée par l'horloge *Clk*. Un décorrélateur de type Von-Neumann est appliqué sur le signal DAS. Avant d'être utilisée, la sortie du TRNG est transformée par une fonction de hachage, le SHA-1. D'après [JK99], le débit du générateur est égal à 75 kb/s ce qui reste un débit très faible.

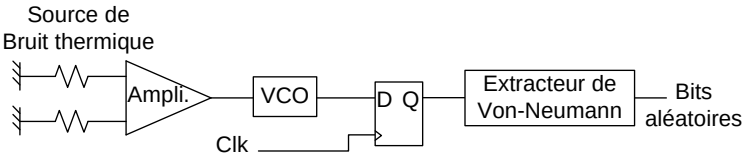

Figure 1-4 – Schéma de principe du TRNG [JK99]. Le bruit thermique du générateur est obtenu en soustrayant les signaux relevés en sortie de deux résistances. La différence des deux signaux impliquant un bruit thermique est ensuite amplifiée. Le signal obtenu est appliqué en entrée d'un oscillateur contôlé en tension (VCO) puis échantillonnée par l'horloge Clk.

Ces circuits utilisant des sources de bruits analogiques sont difficilement implantables au sein des circuits reconfigurables. De plus, le faible débit de ces TRNG limite l'utilisation d'une technique de type amplification directe.

3.2 TRNG basés sur l'échantillonnage d'oscillateurs

Un échantillonnage d'oscillateurs (OS) est basé sur l'utilisation du bruit de gigue généré par un oscillateur. Comme le présente la figure 1-5, le principe est de réaliser l'échantillonnage d'une horloge haute fréquence (F_h) par une horloge basse fréquence (F_l). La gigue dépend de la technique d'oscillateur utilisée. Plusieurs solutions sont possibles. L'horloge touchée par la gigue peut être produite à partir d'une horloge externe ou d'un oscillateur interne au circuit (FPGA ou ASIC). Cette partie présente les OS implantés au sein des circuits VLSI et FPGA.

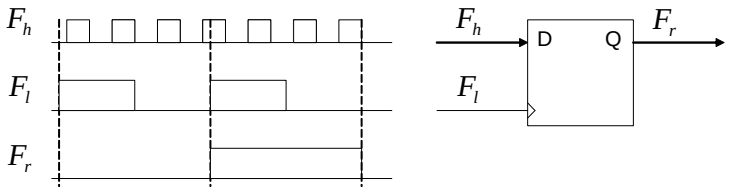

Figure 1-5 – Fonctionnement d'un OS selon [TLL03]. Le principe est de réaliser l'échantillonnage d'une horloge haute fréquence (F_h) par une horloge basse fréquence (F_l).

3.2-1 TRNG proposé par Tsoi et al.

Tsoi et al. [TLL03] présentent l'implantation d'un OS au sein d'un FPGA. Le signal F_l est créé à partir d'un oscillateur externe au circuit. L'oscillateur permet d'augmenter

la gigue présente sur F_l. L'échantillonnage est réalisé au sein du FPGA. Le schéma de principe du générateur est donné dans la figure 1-5. Comme le montre la figure 1-6, afin de diminuer le biais présent sur la sortie générée, une porte XOR est utilisée (*xor extractor*). Le circuit de décorrélation engendre une forte diminution du débit du TRNG. D'après [TLL03], certains paramètres peuvent dégrader grandement la qualité de la séquence issue du TRNG. Un choix judicieux de la fréquence du signal (F_l) doit être réalisé. De plus, une trop forte diminution de la gigue présente sur F_l engendre une forte corrélation. Ceci peut entraîner la prédictibilité du générateur. Enfin, le rapport entre les périodes des signaux F_l (T_{F_l}) et F_h (T_{F_l}) doit être choisi correctement ($T_{F_l} \geq 0.75 T_{F_h}$). Le TRNG fonctionne à une fréquence maximale de 266 MHz (les fréquences respectives de F_l et F_h sont égales à 266 MHz et à 400 MHz). Le TRNG possède un débit de 29 kb/s. L'inconvénient du générateur est la nécessité d'utiliser un circuit externe au FPGA, diminuant par conséquent la sécurité du système.

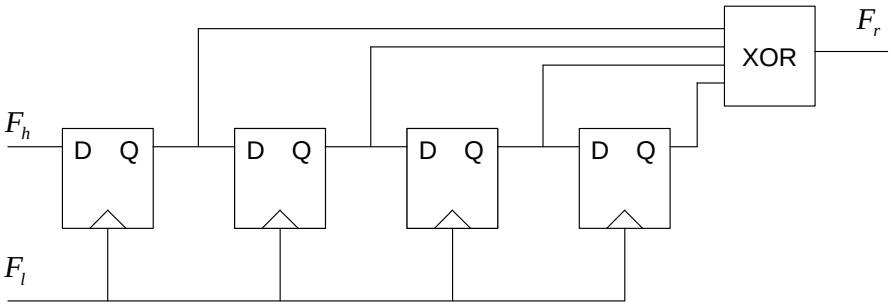

Figure 1-6 – Implantation de l'OS selon [TLL03]. Décorrélateur utilisé par le générateur.

Au sein des circuits cryptographiques, l'utilisation d'une horloge externe est à éviter. L'objectif est de minimiser les possibilités d'attaques sur le TRNG. Une des solutions proposée afin de corriger cette faiblesse est d'utiliser un oscillateur interne au circuit reconfigurable.

3.2-2 TRNG proposé par Fischer et al.

Fischer et al. [FDvB04] présentent un OS utilisant la gigue créée par une boucle à verrouillage de phase (PLL). Ce générateur, introduit dans [FD03] est un des premiers TRNG à avoir été implanté dans une cible FPGA. Plusieurs implantations de ce type ont ensuite été rapportées dans la littérature (FPGA Altera, FPGA Actel).

Le principe de fonctionnement du générateur est de traquer la gigue relative entre deux signaux d'horloges. A la différence de [TLL03], [FDvB04] n'utilise pas nécessairement une horloge basse fréquence. Dans [FDvB04], les deux signaux d'horloges sont générés à partir de blocs PLL (boucle à verrouillage de phase). La variation de bruit dans une PLL engendre

une variation de la fréquence du signal généré. Un circuit de contrôle asservit la fréquence du signal généré à la fréquence désirée. Cette variation représente la gigue. Le schéma de principe du générateur est donné dans la figure 1-7.

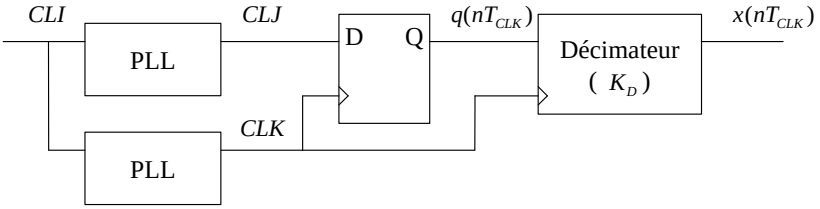

Figure 1-7 – Implantation de l'OS à partir de deux PLL, selon [FDvB04]. Le principe de fonctionnement du générateur est de traquer la gigue relative entre deux signaux d'horloges. Les deux signaux d'horloges sont générés à partir de blocs PLL. La variation de bruit dans une PLL engendre une variation de la fréquence du signal généré. Cette variation représente la gigue.

La gigue se situe autour des fronts montants et descendants du signal CLJ. Les fréquences des signaux CLK et CLJ sont obtenues à partir des relations

$$F_{CLK} = \frac{M_{CLK}}{D_{CLK}} F_{CLI} \tag{1-7}$$

$$F_{CLJ} = \frac{M_{CLJ}}{D_{CLJ}} F_{CLI} = \frac{M_{CLJ} D_{CLK}}{D_{CLJ} M_{CLK}} F_{CLK} = \frac{K_M}{K_D} F_{CLK}. \tag{1-8}$$

où M_{CLK}, D_{CLK}, M_{CLJ} et D_{CLJ} désignent les paramètres des PLL.

Pendant une période du générateur T, K_D fronts montants de CLK et $2K_M$ fronts montants et descendants de CLJ surviennent. Ainsi, K_D valeurs de CLJ sont échantillonnées et au moins une de celles-ci doit être affectée par la gigue. Si T_{CLK} désigne la période de l'horloge CLK et T_{CLJ} celle de l'horloge CLJ, la période du générateur est donnée par

$$T = K_D T_{CLK} = K_M T_{CLJ}. \tag{1-9}$$

Deux conditions doivent être remplies pour assurer le caractère aléatoire de la séquence générée. Les paramètres K_M et K_D doivent être premiers entre eux pour assurer une répartition uniforme des bits en sortie du générateur. Par conséquent, $pgcd(K_M, K_D) = 1$, où *pgcd* désigne le plus grand diviseur commun. De plus, pendant T, au moins un front montant de CLK doit échantillonner le signal CLJ pendant un de ses fronts montants ou descendants (moment présentant l'effet de gigue). D'après [FDvB04], dans le pire des cas, la durée minimale entre le front montant de CLK et un des fronts de CLJ est égale à

$$\max(\triangle T_{min}) = \frac{T_{CLK}}{4K_M} pgcd(2K_M, K_D) = \frac{T_{CLJ}}{4K_D} pgcd(2K_M, K_D). \tag{1-10}$$

En choisissant les paramètres K_M, K_D et F_{CLJ} à partir de (1-10), il est possible de borner la valeur de $\max(\triangle T_{min})$ par l'écart type de la gigue de l'oscillateur σ_{jit}

$$\sigma_{jit} > \max(\triangle T_{min}). \tag{1-11}$$

Le rôle de $\max(\triangle T_{min})$ est présenté à la figure 1-8.

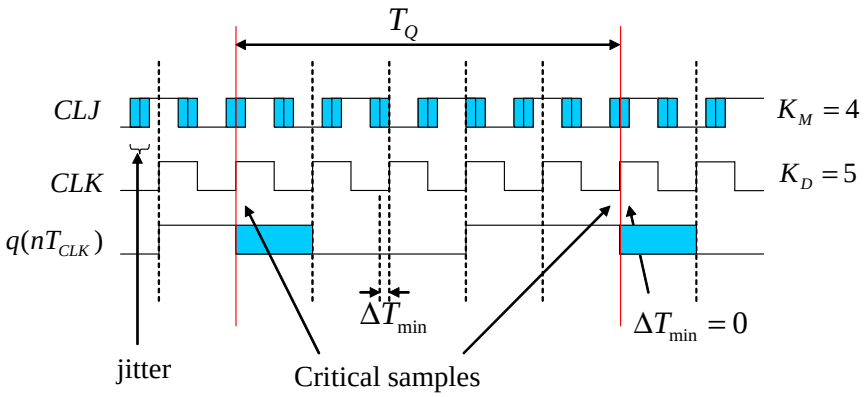

Figure 1-8 – Observation d'un échantillonnage critique dans [FDvB04].

Le bit de sortie du générateur $x(nT)$ est déduit des K_D valeurs échantillonnées de CLJ. Afin de réduire la corrélation présente en sortie du TRNG, [FDvB04] utilise un décimateur basé sur l'utilisation d'une porte XOR

$$x(nT) = q(nT) \oplus q(nT - T_{CLK}) \oplus ... \oplus q(nT - (K_D - 1)T_{CLK}), \tag{1-12}$$

où $\oplus$ désigne l'opérateur XOR et $q(nT)$ désigne l'échantillonnage du signal CLJ par le signal CLK.

[FDvB04] étudie la variation des performances de l'OS lorsque les deux horloges sont générées à partir de différentes PLL (*Enhanced* PLL (EPLL) et *Fast* PLL (FPLL)). La meilleure configuration est obtenue lorsque deux EPLL sont utilisées.

De plus, [FDvB04] montre que si (1-11) n'est pas satisfaite, la qualité du TRNG peut être améliorée en ajoutant une ligne de retard. Ceci est présenté dans la figure 1-9. Le correcteur basé sur l'opérateur XOR permet d'augmenter la probabilité de recouvrement entre le front montant de CLK et les fronts de CLJ.

L'utilisation d'une ligne de propagation permet de rendre équivalente toutes les configurations. Par ailleurs, [FDvB04] montre qu'une plus longue ligne de propagation permet de diminuer le biais du générateur.

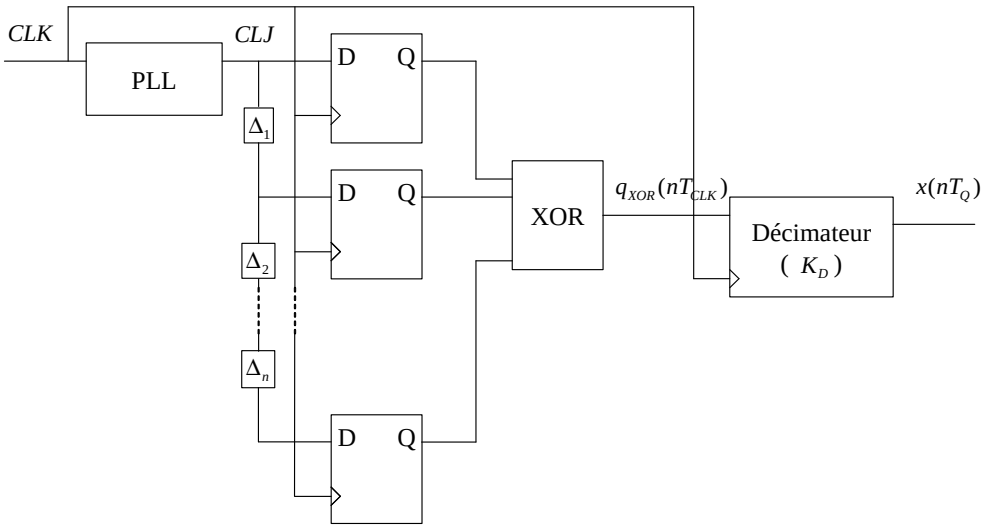

Figure 1-9 – Mise en place de la ligne de retard dans [FDvB04]. Echantillonnage du signal CLJ retardé par plusieurs délais puis combinaison des sorties par une porte XOR.

À partir de (1-11), [FDvB04] montre la dépendance entre le débit du TRNG et la gigue présente

$$F_{CLI} \max(\triangle T_{min}) = \frac{1}{4M_{CLK}M_{CLJ}}, \tag{1-13}$$

où F_{CLI} désigne la fréquence du signal d'entrée. Pour la fréquence de F_{CLI} fixée, minimiser $\max(\triangle T_{min})$ revient à maximiser M_{CLK} et M_{CLJ}. Le débit de l'OS est défini par

$$R = \frac{F_{CLI}}{D_{CLK}D_{CLJ}}. \tag{1-14}$$

Par conséquent, augmenter le débit revient à minimiser D_{CLK} et D_{CLJ}. [FDvB04] a implanté le générateur au sein d'une carte de développement Altera NIOS APEX20K200. Les paramètres du système sont $K_M = 785$, $K_D = 1272$ et $\max(\triangle T_{min}) = \frac{T_{CLK}}{4K_M} = 7.2$ ps $< \sigma_{jit}$. [FDvB04] a aussi implanté le générateur sur une carte Altera Stratix EP1S25. Un débit maximal de 1 Mb/s est obtenu.

[SFDF06] propose un modèle probabiliste permettant de prédire le biais produit en sortie du TRNG [FDvB04]. Le modèle est basé sur la connaissance du nombre de bits aléatoires dans les K_D bits récoltés avant l'étage de post-traitement. Il repose sur l'indépendance entre les K_D bits. Malheureusement, en fonction de la configuration des horloges CLK et CLJ, le modèle n'est pas toujours valide.

Une alternative à l'utilisation de boucles à verrouillage de phase est d'utiliser un ou plusieurs oscillateurs à anneaux.

L'avantage des OS utilisant une technologie à base d'oscillateurs à anneau (RO) est la portabilité de l'implantation dans les circuits VLSI ou FPGA. Un oscillateur à anneau est constitué d'un nombre impair d'inverseurs connectés en série. La sortie du dernier inverseur

reboucle sur l'entrée du premier. Un signal d'horloge dont la période est proportionnelle au nombre d'inverseurs ainsi qu'à leur temps de traversée est ainsi obtenu. L'inconvénient des RO est leur sensibilité aux bruits extérieurs (température, perturbations électromagnétiques, bruits d'alimentation), à l'activité avoisinante et au taux d'occupation du circuit.

3.2-3 TRNG proposé par Bucci et al.

A partir de la conception d'un oscillateur *full-custom* incluant une source de bruit, le TRNG présenté par Bucci et al. [BGL+03] possède un débit de 10 Mb/s et satisfait les tests statistiques du FIPS 140-2. La figure 1-10 présente le fonctionnement du TRNG. La fréquence de l'oscillateur rapide (CLK_{FAST}) est égale à 1 GHz. L'horloge CLK_{JIT} possède une fréquence moyenne de 10 MHz. La gigue présente sur le signal CLK_{JIT} est égale à un dixième de sa période d'horloge. La variance de la gigue de CLK_{JIT} est supposée être supérieure à celle du signal haute fréquence. L'hypothèse permet de supposer la non corrélation des échantillons générés par le TRNG. Dans la figure 1-10, le facteur N_{PRE} est utiliser afin de tester plusieurs débits. Par ailleurs, d'après [BGL+03], le facteur pourrait être adapté à partir de l'évaluation de la qualité du générateur par l'étage de post-traitement.

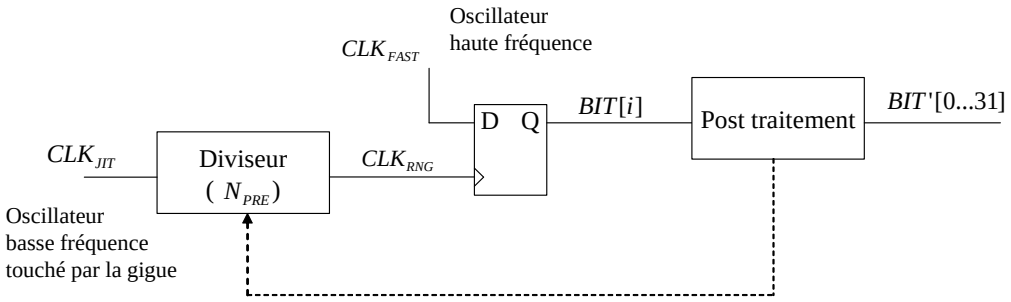

Figure 1-10 – Principe de fonctionnement de l'OS selon [BGL+03]. L'oscillateur rapide (CLK_{FAST}) est implanté par un RO comportant 10 inverseurs, sa fréquence est de 1 GHz. L'horloge CLK_{JIT} possède une fréquence moyenne de 10 MHz. Le post-traitement permet de contrôler le facteur N_{PRE} afin d'adapter la fréquence du signal CLK_{RNG}.

Afin d'analyser le comportement statistique de la source de bruit, [BGL+03] pose quatre hypothèses. Comme présenté dans la figure 1-11, la période du signal CLK_{RNG} ($T_{CLK_{RNG}}$) est supposée suivre une loi normale. De plus, la gigue présente sur l'horloge haute fréquence est négligée. La relation

$$N = \frac{E[T_{CLK_{RNG}}]}{T_{CLK_{FAST}}} \in \mathbb{N}, \tag{1-15}$$

est supposée vérifiée. Dans (1-15), $T_{CLK_{FAST}}$ et E désignent respectivement la période du signal CLK_{FAST} et l'espérance mathématique. Enfin, la différence de phase (t_0) entre les

signaux CLK_{FAST} et CLK_{RNG} appartient à l'intervalle $[0, T_{CLK_{FAST}}]$ (voir la figure 1-11).

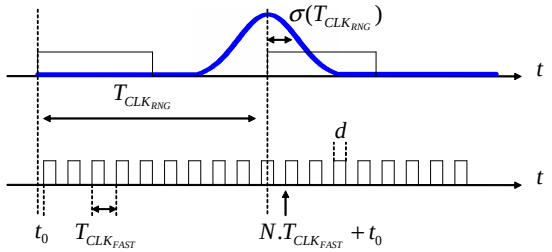

Figure 1-11 – Signal en sortie des oscillateurs selon [BGL+03].

À partir des hypothèses posées, [BGL+03] exprime la probabilité de transition du signal $BIT[i]$ par

$$P_t = P[BIT[i] \neq BIT[i-1]]. \tag{1-16}$$

Dans un générateur idéal, la probabilité de transition est égale à 1/2. A partir de la figure 1-11, il est possible de déterminer la probabilité de transition sur le signal $BIT[i]$.

$$P_t = \sum_{j=-\infty}^{+\infty} \int_{(N+j)T_{CLK_{FAST}}+t_0}^{(N+j+d)T_{CLK_{FAST}}+t_0} p(T_{CLK_{RNG}}) dT_{CLK_{RNG}}, \tag{1-17}$$

où $d \in [d_{min}, d_{max}]$ est la durée pendant laquelle l'horloge CLK_{FAST} est à l'état haut et $p(T_{CLK_{RNG}})$ désigne la densité de probabilité de la gigue présente sur le signal CLK_{RNG}. D'après [BGL+03], la somme infinie dans (1-17) est approximée par une somme finie dans laquelle $j \in [j_{min}, j_{max}]$ avec $j_{max} T_{CLK_{RNG}} \geq 3\sigma_{T_{CLK_{RNG}}}$ où $\sigma_{T_{CLK_{RNG}}}$ correspond à l'écart type de la gigue présente sur le signal $T_{CLK_{RNG}}$.

Le générateur réussit les tests statistiques du FIPS 140-2. Une critique du générateur est qu'il néglige la gigue présente dans l'horloge rapide. L'horloge est générée à partir d'un oscillateur à anneau. Les RO sont pourtant connus pour leurs gigues importantes (gigue aléatoire et gigue déterministe). De plus, [BGL+03] néglige le glissement entre les deux signaux d'horloges engendrés par la gigue. Une dernière critique de [BGL+03] est à propos de la base mathématique du TRNG. La construction du générateur est basée sur l'expression de la probabilité de transition P_t. P_t est approximée par une somme finie de termes dépendants du paramètre d'écart type de l'horloge CLK ($\sigma_{T_{CLK_{RNG}}}$). Dans des conditions réelles d'utilisation, [BGL+03] ne tient pas compte de la variation de ce paramètre. L'approximation de P_t peut devenir non valide en raison de la variation de la gigue.

3.2-4 TRNG PROPOSÉ PAR KOHLBRENNER ET AL.

Une variante de TRNG utilisant des oscillateurs à anneau est présentée par Kohlbrenner et al. [KG04]. Le principe du circuit est toujours le même. Deux horloges sont fournies

par deux RO. Les périodes des deux horloges sont légèrement différentes. La figure 1-12 présente le fonctionnement du TRNG. Les deux oscillateurs à anneau sont implantés selon la structure donnée dans la figure 1-13 qui se compose de deux bascules D dont l'entrée *enable* est égale à 1, d'un tampon et d'un inverseur.

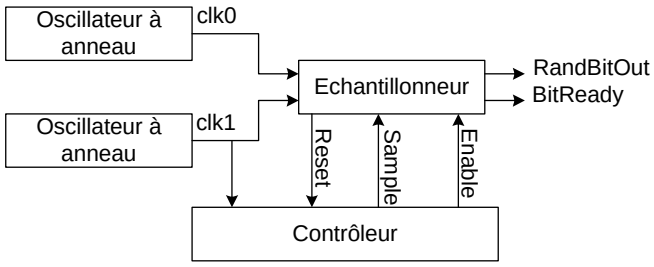

Figure 1-12 – Schéma de principe du TRNG selon [KG04]. Deux horloges sont fournies par deux RO. Les périodes des deux horloges sont légèrement différentes.

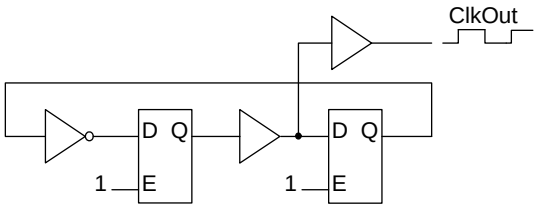

Figure 1-13 – Schéma de l'oscillateur à anneau selon [KG04].

Le bloc échantillonneur donné dans la figure 1-14 a pour objectif d'extraire la gigue présente sur les deux horloges issues des RO. Les signaux $R0$, $S0$ et $E0$ sont des signaux de contrôle. Le chronogramme fourni à la figure 1-15 illustre le fonctionnement de l'échantillonnage. Lors de la présence de gigue sur le signal d'horloge $Clk0$, le signal $S0$ est modifié. La gigue est traquée à partir du signal $C0$. $C0$ réalise le comptage modulo 2 du nombre de cycles pendant lequel le signal $S0$ est à l'état haut. Une variation de la gigue fait varier $S0$ et par conséquent la valeur du compteur. A chaque front montant du signal $S0$, un nouveau bit est mis en sortie, le bit *BitReady* passe à l'état haut et le compteur $C0$ est réinitialisé à 0. Au cours du fonctionnement, il peut arriver que les fronts des horloges $Clk0$ et $Clk1$ deviennent très proches, générant ainsi en sortie du TRNG des bits fortement corrélés. Le bloc contrôleur dans la figure 1-12 permet de détecter cet aléa de fonctionnement. Si le signal $S0$ est à l'état haut pendant un faible nombre de cycles d'horloge, les bits aléatoires ne sont pas mis en sortie. De même, le circuit contrôleur permet de détecter lorsque les fronts des deux horloges sont trop éloignés.

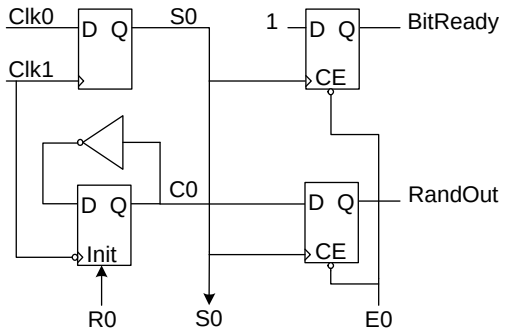

Figure 1-14 – Schéma de principe du bloc échantillonneur de [KG04]. Les signaux $R0$, $S0$ et $E0$ sont des signaux de contrôle. Le circuit permet d'extraire la gigue présente sur les deux horloges.

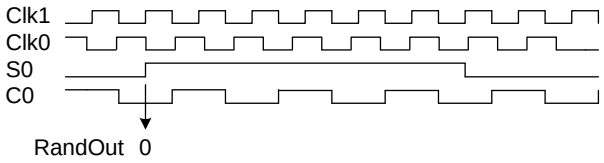

Figure 1-15 – Chronogramme de fonctionnement de l'échantillonneur de [KG04].

L'inconvénient de ce TRNG est que le fonctionnement dépend fortement de l'implantation. La fréquence du générateur dépend des délais de propagation à travers les portes des RO ainsi que des délais de routage introduits par les boucles de retour. La conception de ce circuit impose de prédire les deux grandeurs. Le délai de propagation au travers des portes du RO peut être approximé à partir de la technologie utilisée. Ceci est réalisable pour des ASIC mais devient difficile dans les circuits FPGA où peu d'informations sont fournies par les constructeurs. De plus, le délai de routage est difficilement prédictible au sein des FPGA car il dépend de l'implantation du RO mais aussi de la taille du circuit.

Dans [KG04], le RO est implanté dans un bloc logique programmable (*Configurable Logic Block* (CLB)). En fonction du CLB utilisé, la fréquence du RO varie de 7% [KG04]. La boucle de retour du RO est réalisée en connectant la sortie du CLB à sa propre entrée. D'après [KG04], il est difficile de mesurer la fréquence d'horloge des RO car celle-ci dépend fortement de la température ainsi que de l'activité du circuit autour des CLB. Par conséquent, les deux RO sont routés l'un à coté de l'autre. Afin de diminuer la corrélation, [KG04] modifie le schéma présenté dans la figure 1-12 par un montage utilisant quatre oscillateurs à anneau où ceux-ci sont échantillonnés par un cinquième RO. A chaque cycle, un décorrélateur basé sur une porte XOR est réalisé sur les sorties des quatre RO. De plus,

afin d'éviter une métastabilité sur le signal $S0$, [KG04] insère une bascule D en sortie de ce signal. La séquence générée est biaisée. Par conséquent, un décorrélateur XOR est utilisé. Les bits issus du TRNG sont regroupés par paquets de deux bits. Le résultat de l'opération XOR réalisée sur ces bits est mis en sortie. Le TRNG développé est implanté dans un FPGA Xilinx Virtex XCV1000. Le débit obtenu est égal à 295.76 kb/s. Le générateur valide les tests statistiques du NIST.

Un inconvénient de la solution présentée dans [KG04] est la sensibilité aux bruits de *cross-talk*. La boucle de retour du RO peut être soumise à des perturbations externes causées par la forte activité du circuit. Par conséquent, si une erreur change l'état du RO, le fonctionnement du générateur peut être fortement dégradé. De plus, le débit du TRNG est faible. Un autre inconvénient est que le TRNG ne prend pas en compte l'évolution de la gigue au cours du temps. Plus les RO fonctionnent, plus la gigue s'accumule. Un effet mémoire engendrant une corrélation en sortie est ainsi obtenu. De plus, le fonctionnement du TRNG dépend fortement de l'implantation et de la technologie utilisée. [KG04] ne fournit pas de modèle générique permettant de porter le TRNG sur une autre cible. Implanter le générateur nécessiterait de faire au préalable des mesures de fréquence des RO ainsi que de leur gigue. Enfin, [KG04] ne donne que peu d'informations sur le comportement du bloc contrôleur présent dans la figure 1-12.

3.2-5 TRNG proposé par Tkacik et al.

Tkacik et al. dans [Tka02] proposent un générateur hybride composé de deux sources aléatoires et de deux générateurs pseudo-aléatoires. La figure 1-16 donne le schéma de principe du générateur. Deux PRNG sont utilisés : un automate cellulaire (CA) ayant 3 voisins et 37 cellules et un registre à décalage à rétroaction linéaire (*Linear Feedback Shift Register*) (LFSR) de longueur 43. Dans l'automate cellulaire, toutes les cellules sauf une suivent la même règle. Le générateur vraiment aléatoire est composé de deux oscillateurs à anneau. L'aléa du générateur est créé par la gigue des oscillateurs.
Cependant, la corrélation générée en sortie n'est pas étudiée. De plus, aucune technique d'extraction n'est réalisée en sortie des RO. Malgré la facilité d'implantation de ce générateur, sa sélection est à déconseiller car aucune étude n'a été faite sur son entropie et aucun débit n'est fourni. Enfin, le mécanisme de sélection des 32 bits n'est pas précisé. D'après [Tka02], le TRNG valide les tests statistiques Diehard. [Dic03] a démontré la faible sécurité du générateur. La source d'entropie, limitée à deux oscillateurs à anneau est trop limitée. De plus, l'utilisation d'un générateur pseudo-aléatoire linéaire rend possible la construction d'un modèle linéaire et sa résolution. À partir de la prédiction des fréquences des deux horloges, [Dic03] montre la prédictibilité du TRNG.

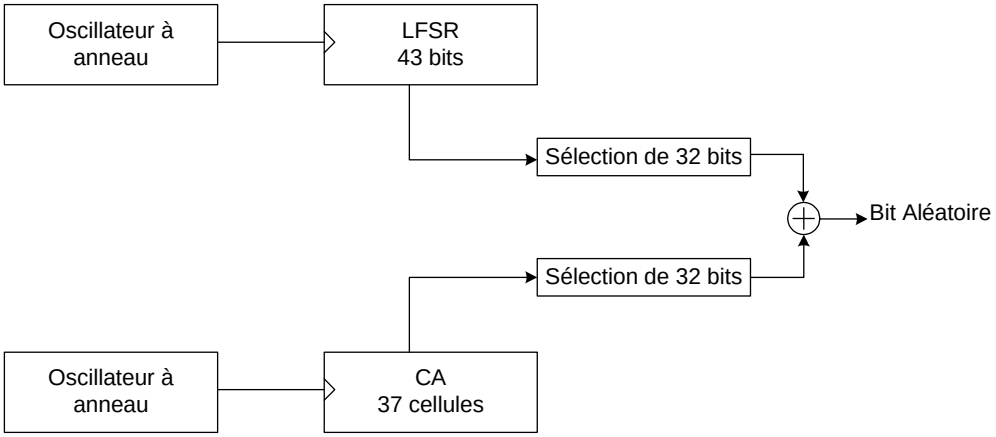

Figure 1-16 – Schéma du générateur hybride de [Tka02]. Le TRNG est composé de deux sources aléatoires (deux RO) et de deux générateurs pseudo-aléatoires, un LFSR et un automate cellulaire).

3.2-6 TRNG proposé par Sunar et al.

Sunar et al. dans [SMS07] présentent un TRNG utilisant l'aléa produit par un grand nombre de RO. La figure 1-17 présente le principe de fonctionnement du générateur. Le TRNG est constitué de 114 RO composés de 13 inverseurs. À partir de l'hypothèse d'une gigue normale $N(\mu, \sigma^2)$, le nombre optimal de RO ainsi que le nombre d'inverseurs requis sont calculés.

Étant donné le nombre trop élevé de RO, le nombre d'oscillateurs minimal permettant de satisfaire un certain critère d'entropie est calculé. La sortie générée par le TRNG est constituée de bits aléatoires et de bits déterministes. Par conséquent, un extracteur d'aléa est utilisé afin de sélectionner les bits aléatoires. Une fonction résiliente (*resilient function*) est choisie puis réalisée en logiciel à partir d'un code BCH (256,16,113). Le débit atteint par le TRNG est de 2.5 Mb/s (à partir d'une fréquence d'échantillonnage de 40 MHz).

Un intérêt de [SMS07] est la modélisation du générateur. L'échantillonnage des oscillateurs est vu comme étant un problème de collecteur de coupons. La période (T) d'un RO est divisée en N urnes. [SMS07] calcule le nombre d'oscillateurs minimum permettant de remplir au moins une fraction des urnes par un bit vraiment aléatoire.

L'inconvénient du TRNG est le nombre d'oscillateurs requis. De plus, [SMS07] affirme que la gigue présente entre chaque RO est indépendante. Pourtant la gigue créée par les RO dépend fortement de la température. Étant donné la proximité des RO, une corrélation entre les différents RO est forcément obtenue.

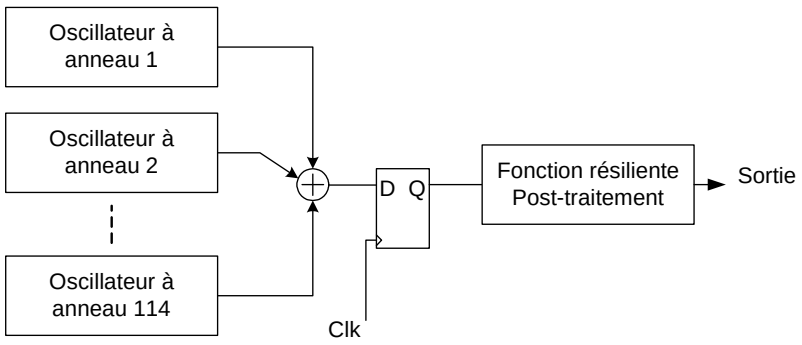

Figure 1-17 – Schéma de principe du TRNG de [SMS07]. Le TRNG est constitué de 114 RO composés de 13 inverseurs. Combinaison des sorties des RO par un arbre de XOR. Une fonction résiliente (*resilient function*) est choisie puis réalisée en logiciel à partir d'un code BCH (256,16,113) pour décorréler la signal DAS.

3.2-7 Améliorations proposées par Schellekens et al.

Schellekens et al. dans [SPV06] ont implanté le TRNG développé dans [SMS07]. Une version minimale du TRNG composé de 110 oscillateurs possédant chacun trois inverseurs est proposée. Mais une solution plus robuste est d'utiliser 210 oscillateurs à trois inverseurs. D'après [SPV06], diminuer le nombre d'inverseurs par oscillateur permet d'augmenter la gigue. Le TRNG réussit les tests statistiques Diehard et NIST.

3.2-8 Améliorations proposées par Yoo et al.

Yoo et al. dans [YSKB] étudient les problèmes de routage liés à la conception des oscillateurs à anneau présents dans le TRNG [SMS07]. De plus, les effets de la variation de la température et de l'alimentation sur le comportement du TRNG sont évalués en simulation. Si la fréquence d'échantillonnage de l'horloge Clk et la fréquence des oscillateurs ne sont pas premiers entre eux, la séquence binaire produite en sortie du TRNG peut être totalement déterministe. En d'autres termes, la condition $pgcd(f_{Clk}, f_{RO}) = 1$ doit être validée avec f_{Clk} et f_{RO} désignant respectivement la fréquence du signal Clk (voir figure 1-17) et celle des oscillateurs à anneau. De plus, [YSKB] montre de manière expérimentale sur cible FPGA que la variation de la température et de la tension d'alimentation du circuit fait varier la fréquence des oscillateurs. Par conséquent, la condition $pgcd(f_{Clk}, f_{RO}) = 1$ n'est pas toujours valide. Ceci présente une opportunité non-négligeable aux attaques non invasives.
Afin d'améliorer la robustesse du TRNG, une architecture utilisant deux longueurs d'oscillateurs à anneau est proposée. Le TRNG valide les tests Diehard et NIST pour un débit de 67 Mb/s. La puissance consommée par le générateur est égale à 300 mW et sa surface est d'environ 1000 LUT sur une cible FPGA Xilinx Virtex II.

3.2-9 TRNG proposé par Bucci et al.

Bucci et al. dans [BGL+06] présentent l'implantation ASIC d'un RNG utilisant des oscillateurs à anneau. La figure 1-18 présente le principe du TRNG. Les particularités de la contribution sont la conception originale des oscillateurs et le fonctionnement du générateur. Les deux horloges sont générées à partir d'un même circuit. La période des horloges est similaire. L'aléa produit par le TRNG est engendré par la gigue relative entre les deux horloges. Lorsque la gigue relative est supérieure à un certain seuil, un bit aléatoire est produit en sortie du TRNG. Un circuit de détection de gigue est donc prévu dans le TRNG. À chaque bit produit, les oscillateurs sont réinitialisés.

Dans [BGL+06], le comportement du TRNG est simulé avec la suite de CAO Cadence en utilisant une technologie 0.12 μm. D'après les simulations, les oscillateurs doivent boucler 100 fois sur eux-mêmes avant d'accumuler une gigue suffisante pour produire un bit aléatoire. Le débit obtenu est en moyenne égal à 13 Mbit/s. Par contre, le TRNG n'a pas été fabriqué et n'a donc pas été validé dans des conditions réelles de fonctionnement ainsi que par des batteries de tests statistiques. Mais la solution de [BGL+06] est très intéressante. Le générateur est asynchrone et produit des bits aléatoires seulement lorsqu'une quantité suffisante de gigue est détectée. Enfin, aucune unité post-traitement n'est requise.

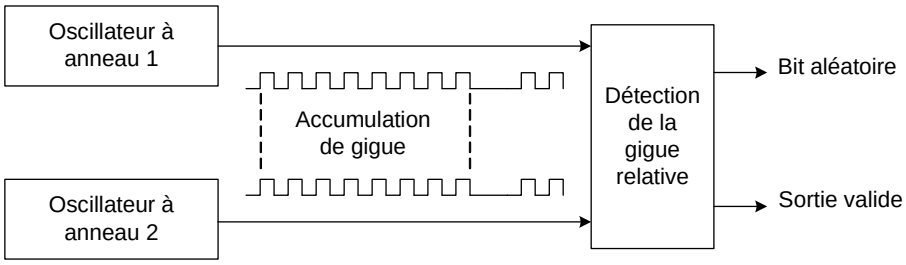

Figure 1-18 – Schéma de principe du TRNG de [BGL+06]. Les deux horloges sont générées à partir de deux RO. L'aléa produit par le TRNG est engendré par la gigue relative entre les deux horloges. Lorsque la gigue relative est supérieure à un certain seuil, un bit aléatoire est produit en sortie du TRNG.

3.2-10 TRNG proposé par Golić.

Golić dans [Gol06] propose une méthode de TRNG basée sur l'utilisation de circuits asynchrones utilisant des boucles de retours. Les circuits mélangent la structure des RO et le comportement des LFSR. Le schéma de principe du TRNG est présenté dans la figure 1-19. Les figures 1-20 et 1-21 correspondent respectivement aux schémas des oscillateurs à anneau de Fibonacci (FIRO) et de Galois (GARO). La structure des oscillateurs est similaire à celle d'un LFSR sauf que les délais sont remplacés par des inverseurs. Les boucles de retour sont contrôlées à partir d'interrupteurs f_i. Si, f_i vaut 1 (respectivement

0), l'interrupteur est fermé (respectivement ouvert). Un FIRO ou un GARO peut être exprimé plus facilement à partir du polynôme formant la boucle de retour

$$f(x) = \sum_{i=0}^{N} f_i x^i, \tag{1-18}$$

avec $f_0 = f_N = 1$.

Les deux oscillateurs fournissent deux signaux périodiques pseudo-aléatoires. Cependant, la période des deux signaux varie en fonction de la gigue induite par les inverseurs. Une addition des deux signaux dans $GF(2)$ est réalisée. Ensuite, la séquence résultante est échantillonnée par un troisième signal d'horloge produit à partir d'un RO. Le circuit de post-traitement est réalisé à partir d'un LFSR. Le TRNG est facilement implantable et valide les tests statistiques Diehard.

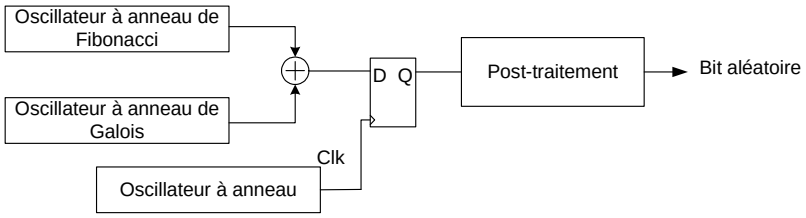

Figure 1-19 – Schéma de principe du TRNG de [Gol06]. Le TRNG est composé d'un oscillateurs à anneau de Fibonacci (FIRO), d'un oscillateurs à anneau de Galois (GARO), d'un RO classique et d'un module de post-traitement (un LFSR).

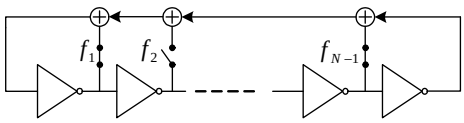

Figure 1-20 – Schéma de principe de l'oscillateur à anneau de Fibonacci selon [Gol06].

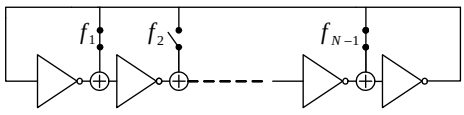

Figure 1-21 – Schéma de principe de l'oscillateur à anneau de Galois selon [Gol06].

L'innovation de [Gol06] est de montrer que la combinaison d'un RO classique avec une boucle de retour plus complexe rend le comportement de l'oscillateur fortement chaotique. De plus, les conditions à satisfaire afin d'éviter qu'un FIRO et GARO ne tombent dans un état stable sont démontrées.

3.2-11 TRNG proposé par Dichtl et al.

Dichtl et al. dans [DG07] étendent les travaux présentés par [Gol06] en étudiant les propriétés statistiques des oscillateurs GARO et FIRO. [DG07] analyse la partie déterministe des oscillateurs. L'étude est menée en réinialisant un grand nombre de fois les oscillateurs dans un même état. [DG07] montre que la partie aléatoire des oscillateurs est plus importante qu'en utilisant un RO classique.

À partir de cette étude, des TRNG utilisant la structure de FIRO et GARO sont présentés. Le TRNG basé sur l'utilisation de GARO est présenté dans la figure 1-22. La sortie du GARO est échantillonnée par une bascule D à la fréquence f_{CLK}. Cependant, pour réduire le biais obtenu, une bascule intermédiaire fonctionnant en mode *toggle* est mise en place entre la sortie du GARO et la bascule D. La bascule intermédiaire compte le nombre de transitions 0-1 effectuées par le signal oscillant du GARO. Le polynôme associé à la boucle de retour utilisé pour le générateur GARO est

$$x^{31}+x^{27}+x^{23}+x^{21}+x^{20}+x^{17}+x^{16}+x^{15}+x^{13}+x^{10}+x^9+x^8+x^6+x^5+x^4+x^3+x+1. \quad (1\text{-}19)$$

Le TRNG fonctionne en mode redémarrage. Chaque fois qu'un nouveau bit aléatoire est produit, le générateur est redémarré dans les mêmes conditions d'initialisation. Sous l'hypothèse qu'après redémarrage, la source aléatoire ne contient pas de terme longuement résiduel (effet mémoire), la source aléatoire échantillonnée est statistiquement indépendante. Cependant, pour confirmer l'hypothèse, le générateur doit être gelé un temps suffisamment long après arrêt de l'oscillateur [DG07]. Le générateur procède en trois étapes. Dans une première phase, le générateur fonctionne comme un RO classique pendant 60 ns. Ensuite, le TRNG est arrêté et les bascules D maintiennent leurs états pendant 40 ns. Enfin, le générateur attend pendant 60 ns avant de regénérer un nouveau bit aléatoire. Le débit du TRNG est égal à 6.25 Mb/s.

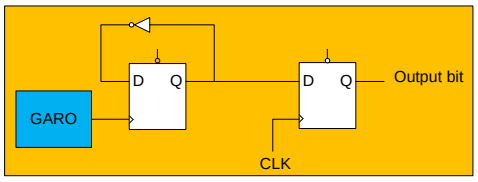

Figure 1-22 – TRNG basé sur l'utilisation de GARO dans [DG07]. Le TRNG est basé sur l'utilisation d'un GARO. Un compteur modulo deux compte le nombre de transitions 0-1 effectuées par le signal oscillant du GARO pendant un cycle de l'horloge CLK. La sortie du GARO est échantillonnée par une bascule D. Le TRNG ne nécessite aucun module de post-traitement.

Le mode redémarrage est intéressant pour réduire la puissance consommée par le générateur car le générateur peut être stoppé lorsqu'aucun germe n'est nécessaire. De plus,

ce mode de fonctionnement est recommandé pour les applications hautement sécurisées [DG07].

3.3 TRNG basés sur la métastabilité d'un signal

Une solution pour générer des nombres vraiment aléatoires est d'utiliser un composant métastable comme source d'entropie. Cette sous partie, décrit le fonctionnement de deux TRNG utilisant ce principe de fonctionnement.

3.3-1 TRNG proposé par Epstein et al.

Le TRNG proposé par Epstein et al. dans [MLR+03] est présenté dans la figure 1-23. Celui-ci est composé de deux multiplexeurs et de deux inverseurs. Le TRNG fonctionne suivant deux phases, une phase d'oscillation et une seconde phase de métastabilité. Si l'entrée *Select* est égale à 0, le TRNG se résume à deux RO classiques. Lorsque *Select* vaut 1, les inverseurs sont mis en cascade et le circuit est stabilisé. La métastabilité du circuit est rencontrée lors d'une transition 0-1 sur le signal *Select*. Dans ce cas les phases des oscillateurs sont non synchronisées et le signal en sortie est imprédictible. Le TRNG est composé de 15 instantiations du circuit présenté dans la figure 1-23. Un opérateur XOR combine les 15 sorties. Après utilisation d'un décorrélateur de Von-Neumann, le TRNG valide les tests Diehard. Cependant, aucune information de performance n'est fourni par [MLR+03].

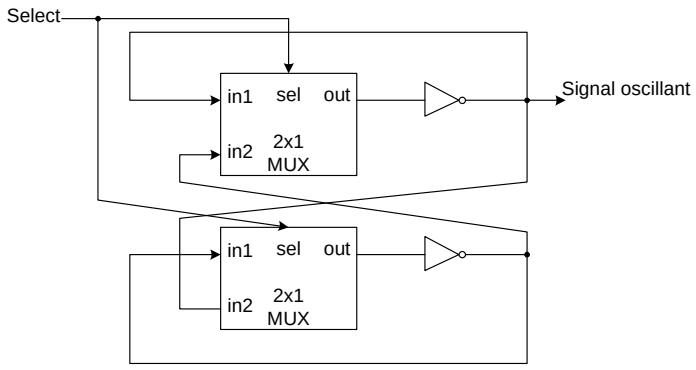

Figure 1-23 – Schéma de principe du TRNG de [MLR+03]. Le TRNG est composé de deux multiplexeurs et de deux inverseurs. Le TRNG fonctionne suivant deux phases, une phase d'oscillation et une seconde phase de métastabilité. Si l'entrée *Select* est égale à 0, le TRNG se résume à deux RO classiques. Lorsque *Select* vaut 1, les inverseurs sont mis en cascade et le circuit est stabilisé. La métastabilité du circuit est rencontrée lors d'une transition 0-1 sur le signal *Select*.

3.3-2 TRNG proposé par Danger et al.

Le principe du TRNG proposé par Danger et al. [DGH07] est présenté à la figure 1-24. Le générateur est basé sur les travaux introduits dans [DGH07]. Le principe de fonctionnement du générateur est d'ajuster la valeur du délai d_1 relativement à d_2 afin que la sortie de la bascule D soit dans un état métastable. Cependant, l'implantation d'un tel circuit présente deux limitations. Premièrement, les bascules D des FPGA sont conçues pour lutter contre les états instables. Si le délai d_2 est réglé de telle manière à violer les temps de conditionnement (temps de *setup*) et de maintien (temps de *hold*), la sortie de la bascule est dans un état métastable. Cependant, la variation de la température, de la tension ou des effets de *crosstalk* engendre une modification de la valeur des délais.

Par conséquent, [DGH09] propose d'utiliser le schéma présenté à la figure 1-25. Le principe du générateur est de former une chaîne de délais permettant de faire varier la valeur de d_1. Les sorties des *latches*, formés par les LUTs, sont échantillonnées afin de détecter les changements en sortie. Le générateur doit satisfaire deux contraintes. Premièrement, la sortie du générateur x ne doit jamais être collée à "1" ou "0". Deuxièmement, le biais engendré par les bruits basse fréquence doit être négligeable.

Lors de l'implantation au sein d'un FPGA Altera EP1S25, les *latches* sont placés manuellement dans 100 cellules connexes regroupées en 10 blocs logiques (*Logic Array Blocks*). Ceci permet de contrôler plus finement les délais de routage. Lors de la variation des paramètres du circuit (tension, température, *crosstalk*), une différence de phase (ΔT) peut exister entre l'horloge (Clk) et les entrées des *latches*. À partir du biais en sortie du TRNG, [DGH09] étudie l'influence du délai (ΔT). Le biais du générateur varie très fortement en fonction de ΔT. [DGH09] conclue qu'il est nécessaire de contrôler et d'ajuster la valeur ΔT afin de réguler le biais du générateur. Le TRNG entier est présenté à la figure 1-26. La chaîne de délai correspond au schéma présenté à la figure 1-25. En sortie, un post-traitement est utilisé afin de diminuer le biais de la séquence générée. [DGH09] utilise un décorrélateur de Von-Neumann. Le débit de sortie du TRNG est égal à 20 Mb/s et satisfait les tests de NIST.

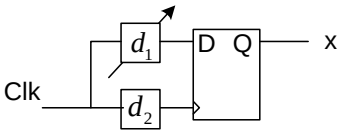

Figure 1-24 – Principe du TRNG proposé par Danger et al. dans [DGH09].

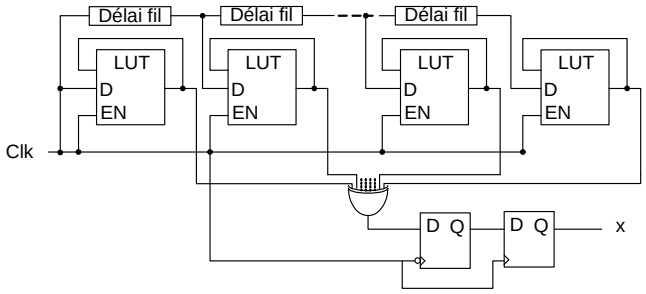

Figure 1-25 – TRNG proposé par Danger et al. dans [DGH09].

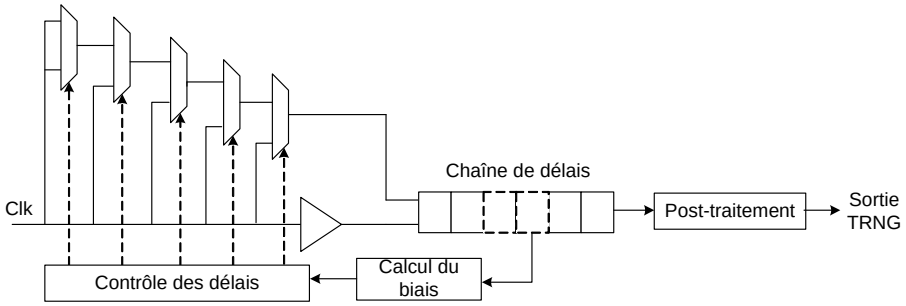

Figure 1-26 – TRNG entier proposé par Danger et al. dans [DGH09]. La chaîne de délai correspond au schéma présenté à la figure 1-25. En sortie, un post-traitement est utilisé afin de diminuer le biais de la séquence générée. [DGH09] utilise un décorrélateur de Von-Neumann.

3.3-3 TRNG proposé par Vasyltsov et al.

Une autre structure innovante a été récemment proposée par Vasyltsov et al. dans [VHKK08]. Son objectif est d'augmenter le débit des TRNG utilisant le principe d'échantillonnage d'oscillateurs. La limitation des TRNG à base de RO classiques est le nombre de cycle requis pour accumuler une quantité de gigue suffisante. Par conséquent, [VHKK08] introduit un TRNG basé sur l'utilisation d'un oscillateur à anneau pouvant être mis efficacement dans un état de métastabilité (META-RO). Le circuit est présenté dans la figure 2-2. La métastabilité permet une accumulation plus rapide de la gigue. Les modes d'opération du TRNG sont présentés dans la figure 2-3. Lors de l'étape de métastabilité (appelée MS), les multiplexeurs déconnectent les inverseurs les uns des autres. Cette étape met chaque inverseur dans un état métastable. Ainsi, les inverseurs forment un ensemble de sources de bruits indépendantes [VHKK08]. Ensuite, dans la seconde étape, les inverseurs sont reconnectés entre eux et forment un oscillateur à anneau classique. Un signal oscillant est produit en sortie de chaque inverseur. Pendant cette étape, le signal oscillant est échantillonné par le signal d'horloge $CLK^{'}$. Afin de réduire le biais en sortie du TRNG, [VHKK08] utilise un extracteur de Von Neumann. La solution proposée par [VHKK08] est composée de cinq étages META-RO.

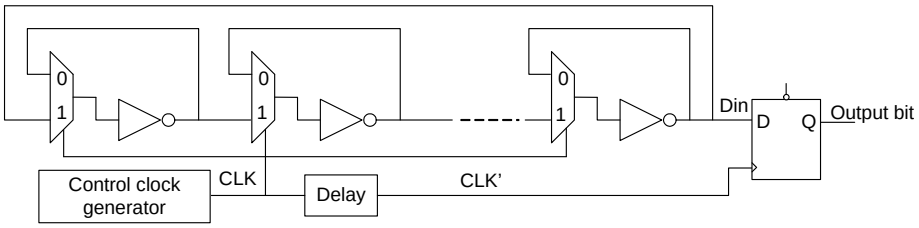

Figure 1-27 – Architecture META-RO proposée par Vasyltsov et al. dans [VHKK08]. Le TRNG est basé sur la possibilité de mettre chaque inverseur dans un état de métastabilité lorsque le signal CLK est égal à 0. Ensuite, lorsque le signal CLK est égal à 1, l'architecture est similaire à un RO classique.

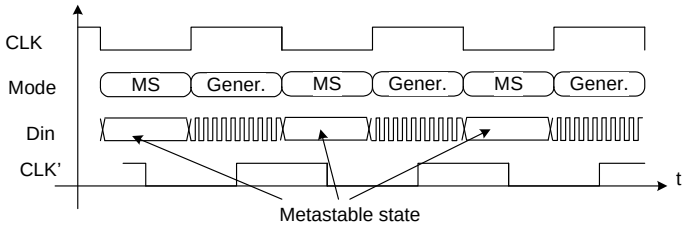

Figure 1-28 – TRNG utilisant le circuit META-RO de [VHKK08]. Lors de l'étape de métastabilité (appelée MS), les multiplexeurs déconnectent les inverseurs les uns des autres. Cette étape met chaque inverseur dans un état métastable. Ensuite, dans la seconde étape, les inverseurs sont reconnectés entre eux et forment un oscillateur à anneau classique. Un signal oscillant est produit en sortie de chaque inverseur. Pendant cette étape, le signal oscillant est échantillonné par le signal d'horloge CLK'.

3.4 Extracteur d'aléas et décorrélateur

Certaines applications nécessitent de disposer de sources parfaitement aléatoires. Les séquences générées par un TRNG sont souvent biaisées et il est par conséquent généralement nécessaire d'extraire les biais aléatoires (*randomness extractor*) ou de décorréler (*deskewing*) la séquence générée. Ces techniques permettent de transformer une séquence aléatoire défectueuse en une séquence quasi-aléatoire indissociable d'une séquence vraiment aléatoire [SV84]. La littérature portant sur les extracteurs d'aléa est très importante. Étonnement, en pratique, les techniques récentes d'extraction d'aléa ne sont pas utilisées [Ist02].

La technique la plus connue est sûrement celle étudiée par Von Neumann. La technique consiste à analyser une séquence binaire en regroupant des paires successives de deux bits. En fonction de la valeur des deux bits, un 0, un 1 logique ou aucun bit n'est généré en sortie du TRNG. Lorsque la séquence '10' (respectivement '01') est observée, un 1 logique

(respectivement un 0 logique) est généré. Par contre, les séquences '11' et '00' ne génèrent aucun bit. La technique de Von Neumann est facile à comprendre. Soit X une séquence déficiente composée de deux symboles possibles $\{0,1\}$. Le biais $|\epsilon|$ de X satisfait la condition $|\epsilon| < 1/2$. La probabilité qu'une observation de X soit un 1 est

$$p = P(X = 1) = \frac{1}{2} + \epsilon. \tag{1-20}$$

Soit Y la variable aléatoire générée par la technique de Von Neumann. La probabilité qu'une observation de Y soit un 1 est donnée par

$$P(Y = 1) = p(1 - p) = \frac{1}{4} - \epsilon^2. \tag{1-21}$$

(1-21) montre que le biais généré en sortie est diminué. De plus, $P(Y = 0) = P(Y = 1)$. L'inconvénient est la réduction importante du débit du TRNG. De plus, l'indépendance des bits biaisés est supposée [BST03].

A partir de la modélisation d'une source défectueuse par une chaîne de Markov finie, [Blu84] montre qu'en changeant l'ordre des bits de la séquence biaisée, une séquence non biaisée et indépendante est obtenue. [Blu84] présente un algorithme efficace générant m bits quasi-aléatoires à partir de n sources défectueuses. A chaque cycle, un bit est extrait des n sources. En contrôlant la parité des n bits recueillis, un bit quasi-aléatoire est généré. Par contre, [Blu84] montre qu'en cas d'attaque engendrant un biais supérieur à un seuil, l'algorithme d'extraction ne fonctionne plus. La séquence quasi-aléatoire générée est aussi mauvaise que la séquence défectueuse.

En pratique, une autre technique répandue est l'utilisation d'une fonction de hachage cryptographique (non universelle).

[NTS99] et [Sha02] présentent un état de l'art des extracteurs d'aléa. Un extracteur déterministe EXT est défini par [Ist02]

$$EXT : \{0,1\}^n \rightarrow \{0,1\}^m. \tag{1-22}$$

L'extracteur extrait l'aléa présent dans un bloc w constitué de n bits en un bloc z de m bits avec $m < n$. w et z sont les réalisations des v.a. W et Z. Un extracteur déterministe ne peut pas extraire parfaitement l'entropie d'une source défectueuse [Ist02].

Un extracteur aléatoire (EXT_r) est défini par

$$EXT_r : \{0,1\}^n \times \{0,1\}^d \rightarrow \{0,1\}^m. \tag{1-23}$$

Dans EXT_r, d bits vraiment aléatoires sont nécessaires.

Soit $P = \{p_0, ..., p_{2^n-1}\}$ et $Q = \{q_0, ..., q_{2^m-1}\}$, les distributions de probabilité des variables aléatoires W et Z. Pour mesurer l'efficacité d'un algorithme d'extraction d'aléas,

deux mesures sont utilisées : la distance statistique et l'entropie minimale [NTS99]. La distance statistique mesure la différence entre deux distributions de probabilité. Par exemple, la distance statistique entre Q et une distribution uniforme est donnée par

$$d(Q,U) = \sum_{x=0}^{2^{m-1}} \left| Q(x) - \frac{1}{2^m} \right| . \tag{1-24}$$

L'entropie minimale de P ($H_{min}(P)$) est donnée par

$$H_{min}(P) = k, \text{ si } P(w = x) \leq 2^{-k}, \ \forall x \in \{0,1\}^n . \tag{1-25}$$

D'après la théorie des extracteurs aléatoires, l'extraction de l'entropie minimale est possible dans une séquence imparfaite.

Dans un extracteur déterministe, la fonction d'extraction (EXT) est souvent réalisée par une fonction de hachage (F). La difficulté est de trouver F (1-23) minimisant la distance statistique entre Q et une distribution uniforme.

Dans un extracteur aléatoire, la fonction de hachage H est choisie aléatoirement

$$H = \{h : \{0,1\}^n \rightarrow \{0,1\}^m\} . \tag{1-26}$$

Dans la distribution de probabilité P de la v.a. W, on posera K le nombre de classes $\in [0, 2^n]$ ayant une probabilité d'apparition différente de 0. La performance d'un algorithme d'extraction dépend de $k = \log_2(K)$ et de la taille du bloc de sortie à générer m [Ist02]. Si $k < m/2$, l'entropie de l'entrée de l'extracteur est pratiquement extraite.

A partir du modèle de l'attaque d'une source aléatoire, [BST03] propose un extracteur simple et efficace. L'extracteur est basé sur l'utilisation d'une fonction de hachage universelle. La fonction de hachage EXT_r est universelle si pour $y \in \{0,1\}^d$, pour $w_1 \neq w_2$, z_1 et z_2, la relation

$$P(EXT_r(w_1) = z_1, EXT_r(w_2) = z_2)) = \frac{1}{2^{2m}} \tag{1-27}$$

est respectée. Si la séquence biaisée possède une entropie minimale suffisante, les images de w_1 et w_2 par EXT_r sont indépendantes et uniformes. Le coût matériel de l'extracteur est comparable à celui engendré par une fonction de hachage cryptographique. De plus, [BST03] montre que n'importe quelle fonction de hachage universelle peut être utilisée. Par conséquent, il est possible de choisir une fonction de hachage universelle efficace tout en assurant la qualité de la transformation.

Lors de la conception d'un extracteur, deux fonctionnements sont envisageables [NTS99]. La première solution est d'extraire tous les bits aléatoires de la séquence biaisée. Ceci implique d'utiliser un nombre de bits vraiment aléatoires élevé. La seconde solution consiste à optimiser le problème en fonction du nombre de bits vraiment aléatoires requis. D'après

[NTS99], la qualité d'un extracteur est fonction du nombre de bits vraiment aléatoires d, de l'entropie de la source défectueuse et du nombre de bits générés m. [NTS99] présente les performances de différents extracteurs reconnus dans la littérature.

En général, l'aléa de la séquence déficiente doit être supérieur à un seuil. [NTS99] propose une solution fonctionnant quelle que soit l'entropie de la séquence biaisée. L'influence des paramètres m et n sur le nombre de bits vraiment aléatoires requis d est étudiée. De plus, l'intérêt de décomposer la séquence biaisée en plusieurs blocs puis d'appliquer une fonction de hachage différente à chacun d'eux est expliqué.

4 Evaluation de la qualité d'un générateur vraiment aléatoire

En pratique l'évaluation d'un TRNG est beaucoup plus complexe que celle d'un PRNG. La qualité d'un TRNG est fonction de la source de bruit utilisée, de l'environnement extérieur au circuit et de la qualité d'implantation au sein du circuit ciblé.

4.1 Méthode d'évaluation d'un générateur pseudo-aléatoire

Prouver mathématiquement qu'un RNG est toujours aléatoire est souvent difficile. Par conséquent, des tests statistiques sont utilisés pour détecter les faiblesses d'une séquence aléatoire x [Knu97]. Un test statistique (T) est un outil de décision analysant une séquence x constituée de N symboles prenant chacun leur valeur dans un ensemble fini, par exemple $B = \{0, 1\}$. T est un algorithme déterministe [Mau91]

$$T : B^N \rightarrow \{H_0, H_1\}, \tag{1-28}$$

où H_0 et H_1 sont les hypothèses du test. H_0 et H_1 désignent respectivement que le RNG est aléatoire et non aléatoire. Le test calcule la probabilité que l'hypothèse H_0 soit acceptée.

Un paramètre important dans un test statistique est l'erreur de première espèce (ou de fausse alarme)

$$\alpha = Prob(H_0 = \text{rejetée} | H_0 \text{ est vraie}). \tag{1-29}$$

α désigne la probabilité de se tromper en rejetant l'hypothèse H_0 à tort, c'est à dire la probabilité que le générateur soit déclaré non aléatoire sachant qu'il est aléatoire. Généralement la valeur de α appartient à l'intervalle $[0.001, 0.05]$.

L'objectif d'un test statistique est de déterminer si une séquence x présente les caractéristiques statistiques d'une séquence aléatoire ou celles d'une séquence non aléatoire. Les générateurs de nombres aléatoires sont testés à partir de différents tests statistiques. Plusieurs batteries de tests statistiques existent dans la littérature. Celles-ci peuvent être fournies par des organismes de standards technologiques comme les batteries NIST [RSNMS01] et

du FIPS 140-2 [Nat99b] ou par des laboratoires de recherche universitaire, par exemple Diehard [MZ93]. [Knu97] explique le fonctionnement de dix tests statistiques différents. [MVO96] décrit plusieurs tests statistiques dont le FIPS 140-2.

Différentes approches sont utilisées par les concepteurs de tests statistiques pour déterminer si une séquence est aléatoire ou non. Les bits présents dans la séquence binaire x doivent être uniformément répartis, indépendants et imprédictibles.

La première approche consiste à effectuer un test statistique sur la séquence binaire x et à interpréter le résultat en fonction d'un certain seuil. Si le résultat est en dessous d'un seuil, le test échoue sinon le test est un succès.

La seconde approche consiste à caractériser le résultat du test en fonction d'un intervalle. Si le résultat du test appartient à l'intervalle alors le test est un succès sinon la séquence binaire x échoue le test. Le test FIPS 140-2 utilise cette technique.

Enfin, la dernière approche consiste à calculer la probabilité P_{value} que la séquence x soit bien aléatoire. On dispose des hypothèses H_0 et H_1. Le test est un échec lorsque $P_{value} < \alpha$, l'hypothèse H_0 est rejetée avec une probabilité d'erreur de α %. Le calcul de la probabilité présente l'avantage de ne pas imposer une valeur de seuil ou un intervalle de confiance pour déterminer si une séquence est aléatoire ou non.

4.2 Evaluation d'un générateur vraiment aléatoire

La méthode de validation mise en place lors de l'évaluation d'un TRNG doit être en relation avec l'application visée. [Sch09] et [Sch01] proposent de classifier les générateurs en quatre classes (R1 à R4). Les deux classes spécifiques aux générateurs cryptographiques sont les classes R1 et R2. La classe R1 impose aux nombres aléatoires générés d'avoir de bonnes propriétés statistiques. La classe R2 implique que la connaissance de séquences de nombres aléatoires ne doit pas permettre de prédire les bits suivants ou précédents en un temps raisonnable et avec une probabilité suffisamment élevée.

La méthode la plus élégante lors de la validation d'un TRNG est de fournir un modèle stochastique de la source de bruit utilisée par le générateur [Sch03] [KS08]. L'objectif des modèles stochastiques est de déterminer l'entropie de la source aléatoire ou au moins une borne minimale de celle-ci. Cependant, la complexité des modèles rend leurs utilisations souvent délicate. Par conséquent, deux batteries de tests statistiques sont très généralement utilisées, le FIPS 140-2 et l'AIS 31.

Le FIPS 140-2 analyse les propriétés statistiques d'une séquence aléatoire de 2×10^4 bits. Elle est composée de quatre tests statistiques. Cependant, la batterie ne fait pas de différence entre l'analyse du bruit échantillonné et du signal généré après post-traitement. L'unité de post-traitement modifie fortement le signal DAS. Par conséquent, certaines défaillances du TRNG peuvent être difficilement détectables.

L'AIS 31 est proposé par l'organisme allemand BSI gérant la validation des TRNG

[AIS01]. L'AIS 31 décompose les générateurs en deux classes, P1 et P2. Les TRNG sont dits de classe P1 (P1-TRNG) si les bits aléatoires internes valident un certain nombre de tests statistiques [KS01]. Cependant, les P1-TRNG ne sont pas suffisants pour les applications telle que la génération de clés aléatoires [SK03]. Par conséquent, la classe P2 (P2-TRNG) a été introduite. Dans les P2-TRNG, le bruit échantillonné est aussi analysé par des tests statistiques. De plus, dépendamment du niveau de sécurité testé, le TRNG doit être testé à l'initialisation et/ou en fonctionnement afin de détecter les défaillances éventuelles de la source de bruit. Par ailleurs, l'AIS 31 conseille aux concepteurs de TRNG de proposer des critères d'évaluation propre à chaque générateur et de fournir des modèles mathématiques de ceux-ci.

5 Conclusions

Après avoir défini le rôle d'un générateur de nombres aléatoires, les applications des RNG ont été présentées. Le rôle essentiel des clés aléatoires en cryptographie a été montré. Par ailleurs, l'importance d'un TRNG de qualité et très performant au sein des méthodes de Monte-Carlo et en communications numériques a été exposée.

Ensuite, un état de l'art des TRNG a été réalisé. Ceux-ci sont basés sur l'amplification d'un bruit physique, l'échantillonnage d'oscillateurs ou la métastabilité d'un signal. La majorité des TRNG implantés au sein de circuits reconfigurables utilisent la gigue d'oscillateurs comme source d'aléa. Cependant, des architectures innovantes combinant échantillonnage d'oscillateurs et métastabilité d'un signal semblent prometteuses.

Dans la littérature, à notre connaissance, aucune comparaison de TRNG entre eux n'a été réalisée. Le tableau 1-1 résume les performances des TRNG à partir des données issues de la littérature. Les débits des générateurs sont relativement faibles, environ quelques Mbits/s. Le TRNG de Yoo et al. [YSKB] possède le débit le plus élevé, soit 67 Mbits/s. Le générateur est une version optimisée du TRNG proposé par Sunar et al [SMS07]. Le TRNG de Vasyltsov et al. [VHKK08] possède lui aussi un débit élevé, entre 35 et 50 Mbits/s sur une cible ASIC. Cependant, peu de détails sont fournis sur les performances obtenues sur une implantation FPGA du générateur. Comparer les performances des générateurs est difficile car ceux-ci sont implantés dans des circuits différents. L'intérêt du chapitre II est de proposer une méthodologie objective analysant les TRNG dans les mêmes conditions d'utilisation.

L'état de l'art des TRNG a montré la forte utilisation des unités de post-traitements au sein des générateurs. Les unités sont utilisées pour corriger les défaillances produites par les TRNG. Certains décorrélateurs très complexes appliquent des modifications importantes sur le signal DAS, par exemple dans les générateurs de Sunar et al. [SMS07] ou de Yoo. et al. [YSKB] utilisant des fonctions de hachage. A la différence, le TRNG proposé par Dichtl et al. [DG07] n'utilise aucun module de post-traitement. Le signal DAS est directement produit en sortie du générateur. Ainsi, la sortie du générateur correspond directement au

bruit échantillonné.

Après avoir détaillé les différents extracteurs d'aléas, un état de l'art des méthodes de validation d'un TRNG a été effectué. Le principe de fonctionnement d'un test statistique a été rapidement présenté. Étant donné l'utilisation des tests statistiques de l'AIS 31 et du FIPS 140-2 dans les chapitres suivants, plus de détails seront fournis dans la suite de ce document.

TRNG	Source de bruit		Débit
	Type source aléa	Source numérique	
Bagini et al. [BB99]	Ampli.	Non	Inc.
Jun et al. [JK99]	Ampli.	Non	75 Kb/s
Tsoi et al. [TLL03]	OS.	Non	29 Kb/s
Fischer et al. [FDvB04]	OS.	Oui	1 Mb/s (FPGA)
Bucci et al. [BGL+03]	OS.	Non	10 Mb/s (simulation)
Kohlbrenner et al. [KG04]	OS.	Oui	295 Kb/s (FPGA)
Tkacik et al. [Tka02]	OS.	Oui	Inc.
Sunar et al. [SMS07]	OS.	Oui	2.5 Mb/s (FPGA)
Yoo et al. [YSKB]	OS.	Oui	67 Mb/s (FPGA)
Bucci et al. [BGL+06]	OS.	Oui	Inc. (simulation)
Dichtl [DG07]	OS.	Oui	6.25 Mb/s (FPGA)
Epstein [MLR+03]	Méta.	Oui	Inc.
Danger [DGH09]	Méta	Oui	20 Mb/s
Vasyltsov [VHKK08]	Méta.	Oui	35 Mb/s - 50 Mb/s (ASIC)

Tableau 1-1 – Comparaison de la performance de TRNG. Dans la seconde colonne, le tableau résume le type de source de bruit utilisé. La colonne trois évalue la possibilité de synthétiser entièrement le TRNG au sein d'un circuit VLSI ou FPGA. Enfin, la colonne quatre présente le débit du générateur. Lorsqu'une information est indisponible dans la littérature, la notation Inc. est reportée dans le tableau.

Chapitre

2 Evaluation de la qualité de générateurs vraiment aléatoires

Comme précédemment introduit, beaucoup d'applications nécessitent d'utiliser un générateur vraiment aléatoire. Récemment, de nouveaux TRNG innovant par leur architecture ont été présentés dans la littérature. L'un d'entre eux possède l'avantage de n'appliquer aucun post-traitement sur le bruit échantillonné. Ainsi, la séquence aléatoire produite par le TRNG dépend seulement de la source de bruit et de son échantillonnage. Choisir un TRNG parmi tous ceux existant dans la littérature peut s'avérer une tâche délicate. A notre connaissance, aucune étude parue dans la littérature compare la qualité des générateurs existants. Il est par conséquent difficile de sélectionner un TRNG. De plus, dans un contexte matériel (i.e. implantation FPGA ou ASIC), la surface et la consommation d'énergie du circuit sont des critères importants nécessitant d'être analysés.

Ce chapitre propose d'évaluer la qualité aléatoire, la surface et la consommation de trois générateurs vraiment aléatoires récemment introduits dans la littérature. Dans la première section du chapitre, les générateurs sélectionnés et leur principe de fonctionnement sont présentés. La seconde section étudie la qualité des générateurs implantés au sein d'un FPGA Altera Stratix II. Le comportement des générateurs en fonction de la température du circuit et de l'activité parasite autour des TRNG est analysé. La troisième section étudie la source de bruit physique des générateurs, la gigue produite par un ou plusieurs oscillateurs. La quatrième section présente une méthodologie d'évaluation des générateurs permettant d'optimiser le débit de ceux-ci. Enfin, la cinquième section est consacrée à l'estimation de la surface et de la consommation des générateurs étudiés.

1 Générateurs vraiment aléatoires sélectionnés

Les générateurs sont sélectionnés pour leur facilité d'implantation au sein de cibles reconfigurables. Les générateurs présentent l'avantage de n'être pas dépendants d'une famille de FPGA. Le TRNG proposé par Ficher et al. [FD03], malgré ses bons résultats n'a

pas été sélectionné car le générateur utilise des PLL indisponibles dans certains FPGA. Récemment, trois TRNG ont été présentés dans la littérature. De par leur structure innovante et les performances présentées dans les publications, ces trois générateurs ont été sélectionnés.

Le **TRNG 1** a été proposé par Sunar et al. dans [SMS07]. Comme le montre la figure 2-1, le générateur est basé sur l'échantillonnage de la gigue de phase d'un grand nombre d'oscillateurs à anneau (RO). Dans [SMS07], chaque RO comprend le même nombre d'inverseurs. Les sorties des RO sont combinées par un arbre de XOR puis échantillonnées à la fréquence $f_{CLK} = 40$ MHz. Les bits échantillonnés sont ensuite utilisés en entrée du code BCH, appliqué comme une fonction résiliente. Un code BCH (pour Bose, Ray-Chaudhuri et Hocquenghem) est un correcteur d'erreur permettant de corriger et détecter un certain nombre d'erreurs dans une séquence binaire. Le TRNG est basé sur le principe selon lequel les transitions sur chaque RO produisent aussi des transitions en sortie de l'arbre de XOR. Par conséquent, si la sortie d'un RO est échantillonnée suffisamment proche d'une transition affectée par la gigue, la séquence binaire résultante est aléatoire. Ici, le TRNG proposé par [SMS07] est composé de 114 RO possédant 13 inverseurs et un code BCH [256,16,113].

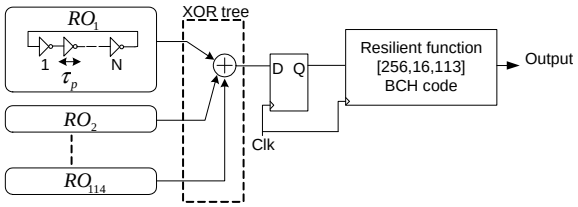

Figure 2-1 – TRNG proposé par Sunar et al. [SMS07]. Le TRNG est composé de 114 oscillateurs à anneau possédant 13 inverseurs. Un arbre de XOR est utilisé afin de combiner les sorties des oscillateurs. Le bloc de post-traitement est effectué par un code BCH [256,16,113].

Le **TRNG 2** est proposé par Vasyltsov et al. dans [VHKK08]. Son objectif est d'augmenter le débit des TRNG utilisant le principe d'échantillonnage d'oscillateurs. Cependant, utiliser des RO classiques n'est pas possible car l'accumulation de gigue est trop lente. Par conséquent, [VHKK08] introduit un TRNG basé sur l'utilisation d'un oscillateur à anneau pouvant être mis efficacement dans un état de métastabilité (META-RO). Le circuit est présenté dans la figure 2-2. La métastabilité permet une accumulation plus rapide de la gigue. Les modes d'opération du TRNG sont présentés dans la figure 2-3. Lors de l'étape de métastabilité (appelée MS), les multiplexeurs déconnectent les inverseurs les uns des autres. Cette étape met chaque inverseur dans un état métastable. Ainsi, les inverseurs forment un ensemble de sources de bruits indépendantes [VHKK08]. Ensuite, dans la seconde étape, les inverseurs sont reconnectés entre eux et forment un oscillateur à anneau classique. Un signal oscillant est produit en sortie de chaque inverseur. Pendant cette étape,

le signal oscillant est échantillonné par le signal d'horloge CLK'. Afin de réduire le biais en sortie du TRNG, [VHKK08] utilise un extracteur de Von Neumann. La solution proposée par [VHKK08] est composée de cinq étages META-RO.

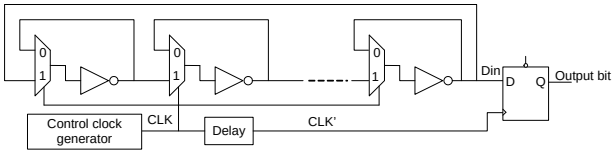

Figure 2-2 – Architecture META-RO proposée par Vasyltsov et al. dans [VHKK08]. Le TRNG est basé sur la possibilité de mettre chaque inverseur dans un état de métastabilité lorsque le signal CLK est égal à 0. Ensuite, lorsque le signal CLK est égal à 1, l'architecture est similaire à un RO classique.

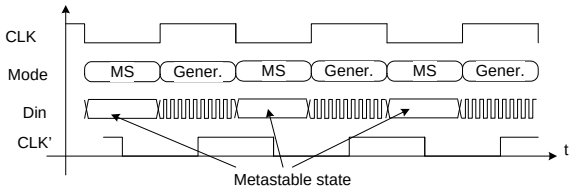

Figure 2-3 – TRNG utilisant le circuit META-RO de [VHKK08]. Lors de l'étape de métastabilité (appelée MS), les multiplexeurs déconnectent les inverseurs les uns des autres. Cette étape met chaque inverseur dans un état métastable. Ensuite, dans la seconde étape, les inverseurs sont reconnectés entre eux et forment un oscillateur à anneau classique. Un signal oscillant est produit en sortie de chaque inverseur. Pendant cette étape, le signal oscillant est échantillonné par le signal d'horloge CLK'.

Le **TRNG 3** utilise le principe des RO de Galois (GARO) introduit par [Gol06]. Pour rappel, un GARO est présenté dans la figure 2-4. Celui-ci est défini par le polynôme caractéristique associé $f(x) = \sum_{i=0}^{r} f_i x^i$, où $f_0 = f_r = 1$.

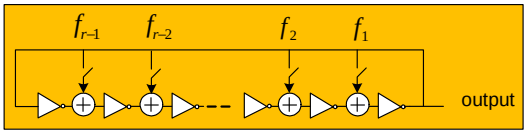

Figure 2-4 – Schéma de principe de l'oscillateur à anneau de Galois selon [Gol06].

[DG07] étend les travaux présentés par [Gol06] en étudiant les propriétés statistiques des GARO et analyse le comportement du GARO dans une configuration spéciale, le mode redémarrage. Chaque fois qu'un nouveau bit aléatoire est produit, le générateur redémarre dans les mêmes conditions initiales. Sous l'hypothèse qu'après redémarrage, la source aléatoire ne contient pas de terme longuement résiduel (effet mémoire), la source aléatoire échantillonnée est statistiquement indépendante. Cependant, pour confirmer l'hypothèse, le générateur doit être gelé un temps suffisamment long après arrêt de l'oscillateur [DG07]. Le mode redémarrage est intéressant pour réduire la puissance consommée par le générateur. De plus, ce mode de fonctionnement est recommandé pour les applications hautement sécurisées [DG07].

Dans cette étude, le **TRNG 3** est le générateur à base de GARO présenté dans la figure 2-5. La sortie du GARO est échantillonnée par une bascule D à la fréquence f_{CLK}. Cependant, pour réduire le biais obtenu, une bascule intermédiaire fonctionnant en mode *toggle* est instanciée entre la sortie du GARO et la bascule D. La bascule intermédiaire compte le nombre de transitions 0-1 effectuées par le signal oscillant du GARO. Le polynôme utilisé pour le générateur GARO est

$$x^{31}+x^{27}+x^{23}+x^{21}+x^{20}+x^{17}+x^{16}+x^{15}+x^{13}+x^{10}+x^{9}+x^{8}+x^{6}+x^{5}+x^{4}+x^{3}+x+1. \quad (2\text{-}1)$$

En mode redémarrage, à chaque génération d'un nouveau bit aléatoire, le générateur procède en trois étapes. Dans une première phase, le générateur fonctionne comme un RO classique pendant 60 ns. Ensuite, le TRNG est arrêté et les bascules D maintiennent leurs états pendant 40 ns. Enfin, le générateur attend pendant 60 ns avant de regénérer un nouveau bit aléatoire. Le débit du TRNG est égal à 6.25 Mb/s. Dans notre étude, la même configuration du générateur est choisie. Afin d'utiliser les mêmes ratio entre les trois phases de fonctionnement, 6 cycles d'horloges ont été prises pour les étapes une et trois et 4 cycles pour la seconde phase.

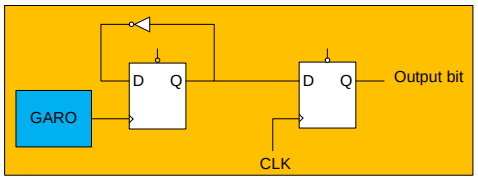

Figure 2-5 – TRNG basé sur l'utilisation de GARO dans [DG07]. Le TRNG est basé sur l'utilisation d'un GARO. Un compteur modulo deux compte le nombre de transitions 0-1 effectuées par le signal oscillant du GARO pendant un cycle de l'horloge CLK. La sortie du GARO est échantillonnée par une bascule D. Le TRNG ne nécessite aucun module de post-traitement.

Les débits des trois TRNG [SMS07], [VHKK08] et [DG07] sont relativement proches. Afin de comparer la qualité de ces générateurs, dans un premier temps, ceux-ci sont paramétrés afin d'obtenir approximativement les mêmes débits, 2.5 Mb/s pour les TRNG [SMS07] et [DG07] et jusqu'à 2.5 Mb/s pour [VHKK08] (en moyenne 1.25 Mb/s). Ces débits sont dans la plage de fonctionnement spécifiée par leurs concepteurs.

Dans un second temps, les générateurs sont implantés et analysés au sein d'un FPGA Altera Stratix II EP2S60. L'objectif est d'étudier la qualité des générateurs lorsque ceux-ci sont utilisés dans les mêmes conditions.

2 Evaluation des générateurs au sein d'un FPGA Altera Stratix II

2.1 Objectif de l'évaluation

L'objectif de cette partie est de proposer une méthode d'évaluation de la qualité des TRNG rapide et efficace. Les TRNG peuvent être influencés par leur environnement extérieur ainsi que par l'activité du FPGA. Par conséquent, la robustesse des TRNG face aux variations de ces deux paramètres doit être évaluée.

La plage de fonctionnement du FPGA Altera Stratix II EP2S60 dans lequel sont implantés les générateurs est de $-40°C$ à $100°C$. Cependant, la température recommandée se situe entre $0°C$ et $55°C$. Dans le cadre de l'étude, les TRNG sont soumis à trois températures différentes : $25°C$ (température ambiante), $55°C$ (température maximale recommandée) et $75°C$.

Par ailleurs, le comportement des TRNG peut varier en fonction de l'activité du circuit. Par conséquent, deux configurations du FPGA sont ici testées. La première consiste à n'appliquer aucune activité à proximité des générateurs. La seconde configuration a pour objectif de tester le comportement des générateurs lorsque ceux-ci sont soumis à une très forte activité environnante. Bien que cette configuration doit être évitée, celle-ci peut être rencontrée en pratique au sein des circuits cryptographiques faibles coûts (i.e. contraintes en surface importantes). Comme le présente la figure 2-6, l'activité autour de la source de bruit des TRNG est réalisée en plaçant des oscillateurs à anneau autour de celle-ci.

Comme introduit dans le chapitre I, l'évaluation de la qualité de générateurs vraiment aléatoires est généralement réalisée via l'utilisation de tests statistiques. Un grand nombre de tests statistiques existent dans la littérature et sont généralement développés en utilisant des langages de programmation haut niveau. Lorsqu'un TRNG est en phase de validation, le concepteur mémorise une grande séquence binaire générée par celui-ci et soumet celle-ci

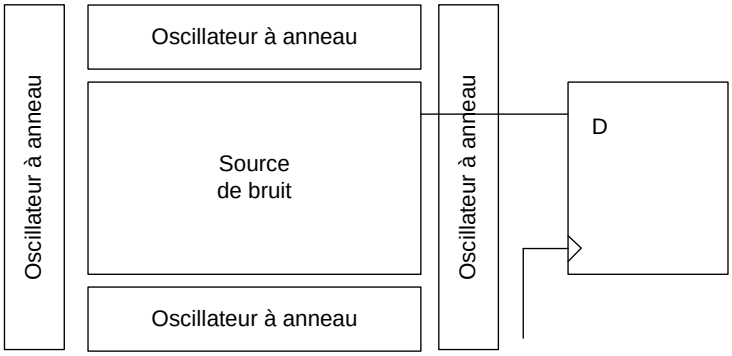

Figure 2-6 – Simulation d'une activité autour de la source de bruit des TRNG en plaçant un grand nombre d'oscillateurs à anneau.

à un certain nombre de tests statistiques. Si la séquence binaire valide un ensemble de tests statistiques (dépendant de l'application visée), le générateur est considéré comme étant suffisamment aléatoire. Les batteries de tests statistiques reconnues sont souvent conçues pour la validation de générateurs pseudo-aléatoires. Malheureusement, la validation de TRNG est plus compliquée à réaliser que celle d'un PRNG. Le comportement d'un TRNG dépend de sa construction, de l'environnement extérieur dans lequel celui-ci est utilisé et essentiellement de la source de bruit physique (pouvant différer en pratique de son comportement idéal). Deux batteries sont essentiellement utilisées en pratique, le FIPS 140-2 et l'AIS 31.

Comme l'illustre la figure 2-7, la méthodologie proposée est d'évaluer la qualité des TRNG grâce à l'utilisation de tests statistiques embarqués, fonctionnant en temps réel sur l'ensemble de la séquence. Ainsi, il est possible en implantant les TRNG au sein du même circuit, d'estimer leur qualité dans des conditions d'utilisation similaires. De plus, le circuit permet d'étudier l'influence de la température du FPGA et de son activité sur le comportement des générateurs. Un autre avantage est de réduire le temps de l'évaluation par rapport à une analyse plus conventionnelle basée sur l'utilisation de tests statistiques logiciels. Les configurations des TRNG (variation de la température et de l'activité du FPGA) à analyser et le nombre de tests statistiques à effectuer rendent nécessaire l'acquisition d'un grand nombre de séquences aléatoires. Par conséquent, une évaluation logicielle est difficilement envisageable car longue et fastidieuse. La méthode permet d'éviter le problème d'interfaçage entre le FPGA et un système extérieur car ici, seuls les résultats des tests statistiques nécessitent d'être transférés. De part l'efficacité de son implantation, le FIPS 140-2 a été choisi pour évaluer la qualité des TRNG sélectionnés.

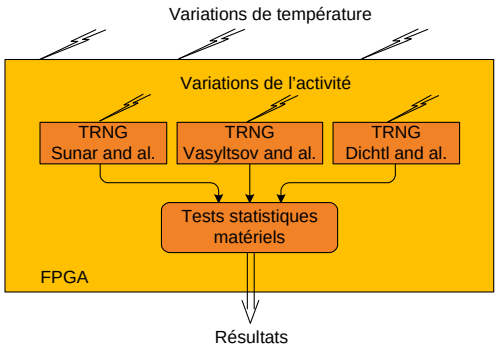

Figure 2-7 – La méthodologie proposée est d'évaluer la qualité des TRNG grâce à l'utilisation de tests statistiques embarqués, fonctionnant en temps réel sur l'ensemble de la séquence.

2.2 Description et implantation de la batterie FIPS 140-2

Le FIPS 140-2 est une batterie composée de quatre tests statistiques, le *frequency* test, le *poker* test, le *run* test et le *long-run* test. Ces tests analysent une séquence aléatoire constituée de 2×10^4 bits.

2.2-1 *Frequency* test

Le test vérifie l'uniformité de la séquence composée de 2×10^4 bits. Le biais de celle-ci est analysé. Le nombre de "1" (n_1) est calculé et doit appartenir à l'intervalle $[9726, 10274]$.

2.2-2 *Poker* test

La séquence binaire est décomposée en séquences indépendantes de quatre bits. Ainsi, 5×10^3 blocs (ou sous-séquences) sont obtenus. Le *poker* test vérifie l'uniformité des sous-séquences. Le nombre d'occurrences des 16 blocs possibles est calculé. Soit n_i, le nombre d'occurrences du bloc d'indice i, avec $i \in \{0, \ldots, 15\}$. Le résultat de X_{15}^2 donné par

$$X_{15}^2 = \frac{2^4}{5 \times 10^3} \left(\sum_{i=0}^{15} n_i^2 \right) - 5 \times 10^3, \tag{2-2}$$

suit un Chi-deux à 15 degrés de liberté et doit appartenir à l'intervalle $]2.16, 46.17[$ pour satisfaire le *poker* test.

En matériel, calculer l'équation (2-2) est une tâche coûteuse en termes de surface et d'unité arithmétique. Une alternative est de calculer le résultat de la sommation $\sum_{i=0}^{15} n_i^2$ et de comparer le résultat avec l'intervalle requis $]Y_{min}, Y_{max}[$. Dans $\sum_{i=0}^{15} n_i^2$, la valeur maximale d'un compteur (n_i) doit être calculée en fonction des valeurs des autres compteurs $(n_j,\ j \neq i)$.

Le problème d'optimisation

$$\begin{array}{ll} max(n_i) & \text{sous contraintes que} \\ & Y_{min} < \sum_{j=0}^{15} n_j^2 < Y_{max} \\ & \sum_{j=0}^{15} n_j = 5 \times 10^3 \\ & n_j \geq 0,\ 0 \leq j \leq 15 \end{array} \tag{2-3}$$

doit être résolu. La valeur de n_i est maximale lorsque la valeur de $\sum_{j=0, j\neq i}^{15} n_j^2$ est minimale et que les contraintes sont satisfaites. La valeur maximale d'un compteur est égale à 428. La résolution du problème d'optimisation a permis de réduire le nombre de bits nécessaires aux compteurs ansi que la taille des opérateurs du graphe flot de calcul car seulement un multiplieur 9×9 et un additionneur 21×21 sont requis.

2.2-3 *Run* test

Le *run* test détermine si le nombre de *runs* (séquence binaire consécutive de "0" ou de "1") de différentes longueurs coïncide avec celui attendu d'une séquence aléatoire. Soit n_i^1 et n_i^0 le nombre respectif de 1-*runs* (séquence consécutive de "1") et de 0-*runs* (séquence consécutive de "0") de longueur i. Pour $i \in \{1, 6\}$, les n_i^1 et n_i^0 sont comptés. Si les occurrences obtenues satisfont le tableau 2-1, la séquence aléatoire valide le *run* test. Seulement douze compteurs sont nécessaires pour mémoriser les occurrences de chaque *run*.

i	Intervalles requis pour n_i^0 et n_i^1
1	2343-2657
3	1135-1365
3	542-708
4	251-373
5	111-201
≥ 6	111-201

Tableau 2-1 – Intervalles requis dans le *run* test.

2.2-4 Long run test

Le test vérifie qu'aucun *run* de longueur 26 bits n'existe. Le test nécessite seulement un compteur de 5 bits.

Cette partie a montré que les tests sélectionnés ne requièrent que très peu de ressources arithmétique et mémoire. Même si les quatre tests statistiques sont peu performants face aux batteries Diehard [Mar96], NIST [RSNMS01] et AIS 31 [KS01], le coût matériel de l'architecture est très intéressant.

2.2-5 Résultats d'implantation pour technologies ASIC et FPGA

Les tests statistiques sont synthétisés par l'outil Synopsys Design Compiler en ciblant une technologie CMOS 130 nm. Le tableau 2-2 résume la surface totale, la puissance dynamique estimée et le délai du chemin critique du circuit. Les résultats de puissance sont obtenus à partir d'une tension d'alimentation de 1.2V.

Surface (mm^2)	Puissance (mW)	Chemin critique (ns)
0.29	7.37	6.08 (164.47 MHz)

Tableau 2-2 – Surface totale, puissance dynamique consommée et chemin critique en utilisant une technologie CMOS 130 nm, 1.2V.

Les tests statistiques sont ensuite synthétisés pour plusieurs cibles FPGA Altera et Xilinx. Le tableau 2-3 montre le nombre total de tables de vérité (LUT) ainsi que la fréquence maximale du circuit au sein des FPGA Xilinx. De plus, le taux d'occupation du composant est présenté. [HKU03] a implanté précédemment le FIPS 140-2 dans une cible Xilinx Virtex 2. La fréquence maximale obtenue par le circuit était de 113 MHz. Comme le montre le tableau 2-3, les résultats de l'implantation réalisée dans cette partie sont meilleurs. Le tableau 2-4 présente les performances du FIPS au sein du FPGA Altera utilisé dans cette partie. Les ALM désignent les modules logiques adaptatifs utilisés dans les FPGA Stratix II. Ceux-ci sont composés d'une LUT à 8 entrées, de deux additionneurs deux bits, de deux multiplexeurs et de deux registres.

2.2-6 Résultats de l'évaluation des TRNG par la batterie FIPS 140-2

Pour chaque configuration des générateurs, 500 réalisations FIPS 140-2 sont effectuées afin d'analyser une séquence binaire de 10 millions de bits aléatoires. Pour des raisons de clarté, pour chaque configuration, le pourcentage de succès du FIPS 140-2 est calculé par

$$\begin{aligned}\text{pourcentage réussite}(\%) = \quad & (\textstyle\sum_{i=1}^{500}(\text{décision frequency test}(i) \times \text{décision poker test}(i) \\ & \times \text{décision run test}(i) \times \text{décision long run test}(i))\end{aligned}$$

FPGA ciblés	Xilinx Virtex 2 XC2V1000-6	Xilinx Virtex 5 XC5VLX50T-3
Nombre total de LUTs	626 (6%)	548 (1%)
Fréquence maximale	134.7 MHz	187.3 MHz

Tableau 2-3 – Résultat de l'implantation du FIPS 140-2 dans les FPGA Xilinx ciblés.

FPGA ciblé	Altera Stratix II EP2S60-3
Nombre ALM	431
Fréquence maximale	166.5 MHz

Tableau 2-4 – Résultat de l'implantation du FIPS 140-2 dans le FPGA Altera Stratix II ciblé.

où les décisions sont égales à 1 (respectivement 0) si le test correspondant est réussi (respectivement échoué) par le TRNG. En d'autres termes, lors d'une réalisation du FIPS 140-2, le TRNG analysé est considéré vraiment aléatoire si et seulement si les quatre tests statistiques sont satisfaits.

Les figures 2-8 et 2-9 résument les taux de réussite du générateur de Sunar et al. [SMS07] aux tests du FIPS 140-2 en absence et en présence d'une forte activité autour du TRNG. Les figures montrent que le générateur est pratiquement toujours aléatoire quel que soit l'environnement dans lequel celui-ci est utilisé. La variation de la température n'a pas d'influence sur les résultats. Par contre, l'activité autour du TRNG diminue très légèrement sa qualité, comme le montrent les taux de succès du *run test*.

Les figures 2-10 et 2-11 montrent les résultats obtenus pour le générateur de Vasyltsov et al. [VHKK08]. La variation de la température du FPGA dégrade légèrement le comportement du générateur. Il en est de même lorsque le générateur est entouré par une forte activité. D'autre part, le *poker test* ainsi que le *run test* sont souvent échoués par le générateur.

Les résultats obtenus par le générateur Dichtl et al. [DG07] sont donnés dans les figures 2-12 et 2-13. Sans activité autour du TRNG, les taux de succès sont relativement bons. Comme le montre la figure 2-12, la variation de la température n'a aucun effet sur le comportement du générateur. Cependant, lorsqu'une forte activité est appliquée autour du TRNG, le taux de succès du *frequency test* est dégradé fortement.

Le TRNG [SMS07] semble être plus aléatoire et robuste que les deux autres générateurs. Pour pratiquement toutes les configurations testées, le pourcentage de succès FIPS 140-2 est proche de 100%.
De plus, l'expérimentation montre que la variation de la température du FPGA n'a que peu d'effet sur le comportement du générateur. Par contre, une forte activité autour des TRNG entraîne généralement une dégradation de leur qualité.

2.2-7 Conclusions de l'évaluation par les tests FIPS 140-2

Les expérimentations montrent que même si un TRNG est satisfaisant (par exemple le générateur de Sunar et al.), sa qualité peut fluctuer au cours du temps. Par conséquent, le générateur peut produire une séquence présentant certaines défaillances, totalement inadmissible dans un contexte de sécurité élevé. De plus, les expérimentations indiquent que

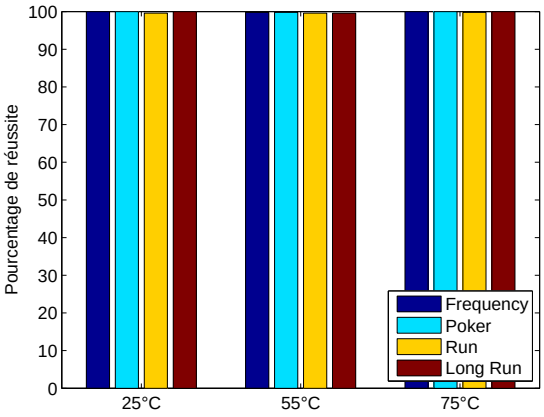

Figure 2-8 – Taux de réussite des tests du FIPS 140-2 sans activité autour du générateur de Sunar et al. [SMS07].

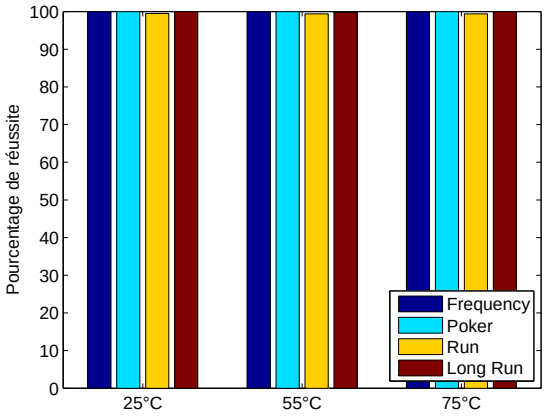

Figure 2-9 – Taux de réussite des tests du FIPS 140-2 avec activité autour du générateur de Sunar et al. [SMS07].

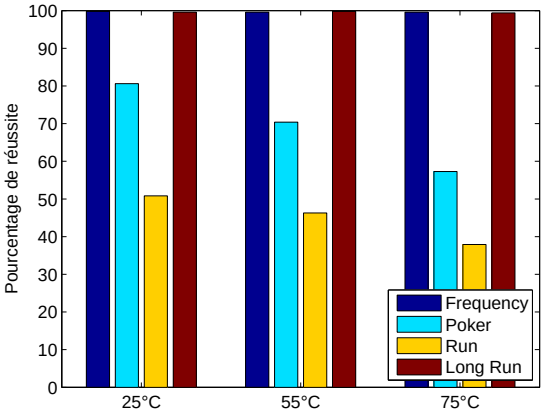

Figure 2-10 – Taux de réussite des tests du FIPS 140-2 sans activité autour du générateur de Vasyltsov et al. [VHKK08].

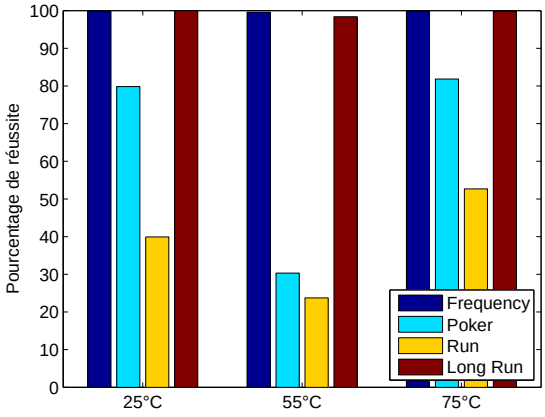

Figure 2-11 – Taux de réussite des tests du FIPS 140-2 avec activité autour du générateur de Vasyltsov et al. [VHKK08].

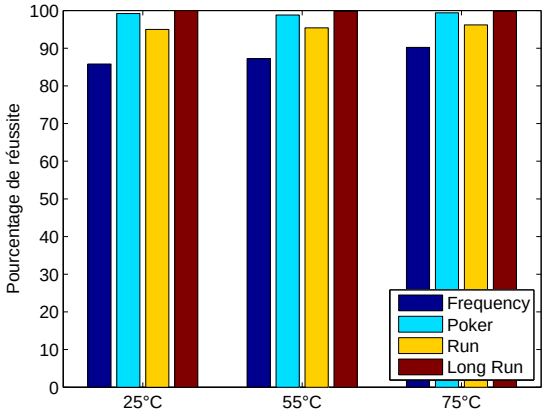

Figure 2-12 – Taux de réussite des tests du FIPS 140-2 sans activité autour du générateur de Dichtl et al. [DG07].

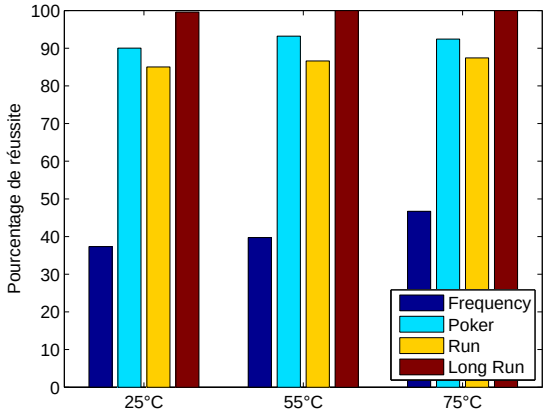

Figure 2-13 – Taux de réussite des tests du FIPS 140-2 avec activité autour du générateur de Dichtl et al. [DG07].

le générateur [SMS07] semble être de meilleure qualité et plus robuste que les deux autres TRNG testés.
Cependant, pour le générateur [DG07], la séquence analysée est la source de bruit échantillonnée. [VHKK08] et [SMS07] appliquent tous deux un post-traitement sur la source de bruit échantillonnée afin de réduire les défaillances produites par les générateurs. [VHKK08] utilise un décorrélateur très simple, l'extracteur de Von Neumann. A la différence, [SMS07] applique une transformation plus compliquée.

Le FIPS 140-2 ne fait aucune différence entre le test d'une séquence issue de l'échantillonnage du bruit ou de la sortie après post-traitement. Malgré les bons résultats obtenus par [SMS07], la complexité de la fonction résiliente (code BCH) peut cacher certaines défaillances du générateur. Par conséquent, l'étude doit être complétée par l'évaluation de la qualité des séquences produites après échantillonnage de la source de bruit.

Une seconde critique de la méthode d'évaluation utilisée est le fait que le débit des TRNG a été fixé afin que celui-ci soit inférieur à ceux spécifiés dans les publications. Le débit des générateurs est fonction de la gigue présente dans le circuit. L'objectif de la partie suivante est de montrer la nécessité d'adapter le débit des générateurs en fonction du circuit utilisé.

Par ailleurs, l'analyse a montré que le générateur [VHKK08] est de moins bonne qualité dans une cible FPGA. Ceci provient du fait que le signal métastable possède un biais non négligeable. En technologie ASIC, il est possible de faire tendre le signal vers $V_{dd}/2$. En technologie FPGA, la métastabilité est plus difficile à obtenir et un biais existe. À notre avis, ceci est dû à l'utilisation de résistance de tirages. Par conséquent, la suite de ce chapitre se consacre à l'analyse des TRNG [SMS07] et [DG07].

3 Variation de la gigue en fonction du FPGA utilisé et de son activité

Les générateurs TRNG [SMS07] et [DG07] sont basés sur la gigue d'un ou plusieurs oscillateurs à anneau. La gigue est la déviation d'un signal par rapport à son comportement idéal. Celle-ci est produite par des effets non désirables tels que le bruit d'alimentation, le bruit thermique ou encore la gigue d'une horloge de référence [Fah05]. La variation de la gigue peut entraîner la défaillance d'un générateur. Par exemple, le nombre de RO composant [SMS07] ainsi que son débit sont fonction de l'écart type de la gigue produite par un oscillateur à anneau. Cependant, en pratique, une mesure fine de cette grandeur est difficile et nécessite l'utilisation d'un matériel onéreux (i.e. sonde active, oscilloscope à bande passante élevée). De plus, mesurer la gigue de manière externe ne correspond pas forcément à la gigue interne du circuit étant donné que celle-ci est fortement modifiée par les ports d'entrées/sorties du circuit.

L'objectif de cette partie est d'évaluer la variation de la gigue en fonction du FPGA utilisé et de son activité. Pour se faire, la gigue produite par un oscillateur à anneau du TRNG [SMS07] est mesurée dans des conditions réelles, au sein de deux FPGA, un Altera Stratix II EP2S60 et un Xilinx Virtex 4 SX35. La mesure sélectionnée est la gigue cycle à cycle. Celle-ci mesure la variation temporelle entre deux périodes d'horloge consécutives. La mesure permet de détecter la gigue déterministe dans le domaine temporel [Fah05]. La gigue cycle à cycle est une mesure efficace pour les générateurs étant donné que le signal oscillant est généralement échantillonné par une horloge propre.

Dans un premier temps, la variation de la gigue en fonction de la température du FPGA a été étudiée. Cependant, les résultats ont montré que la variation de la température n'a aucun impact sur les caractéristiques de gigue. Par conséquent, les résultats obtenus ne sont pas présentés ici. Dans un second temps, la variation de la gigue en fonction de l'activité du circuit est examinée pour une température du FPGA égale à 25°C. Deux configurations sont testées. La première d'entre elles correspond à l'instantiation d'un seul RO au sein du FPGA. Aucune activité néfaste n'est générée autour de l'oscillateur. La seconde configuration étudie l'impact d'une très forte activité autour du RO. Bien que cette configuration doit être évitée en pratique, ceci peut arriver au sein des circuits cryptographiques à bas coût en raison de contraintes en surface. L'activité est créée en instantiant un grand nombre d'oscillateurs autour du RO examiné.

La mesure de gigue est réalisée en utilisant un oscilloscope Agilent DSA91304A Infiniium (13GHz, 40 GSa/s) et une sonde active Agilent 1169A 12 GHz Infinimax. L'analyse de gigue est réalisée sur 20×10^6 échantillons. Afin de réduire le bruit de mesure obtenu avec l'oscilloscope, la bande passante de celui-ci est réduite à 1 GHz. La gigue cycle à cycle est mesurée grâce au logiciel Agilent EZJIT [Tec08]. Les résultats présentés ici contiennent l'histogramme, la tendance et le spectre de la gigue. Pour des raisons de clarté, les figures montrant le spectre utilisent les mêmes échelles (1ps/div pour l'échelle verticale et

6MHz/div pour l'échelle horizontale). Les deux mesures principales sont l'histogramme et le spectre de la gigue. L'histogramme permet d'évaluer la distribution de la gigue. Le spectre est utile pour détecter la partie déterministe de la gigue cycle à cycle, difficile à visualiser dans l'histogramme.

3.1 Mesure de la gigue cycle à cycle

3.1-1 Analyse de la gigue du RO utilisé dans des conditions idéales de fonctionnement

Les résultats de mesure obtenus lors de l'implantation du RO au sein d'un FPGA Xilinx Virtex-4 SX35 et d'un Altera Stratix II EP2S60 sont présentés respectivement dans les figures 2-15 et 2-14.
La figure 2-14 montre qu'au sein du FPGA Altera, l'histogramme suit une loi normale et le spectre est similaire à un bruit blanc.

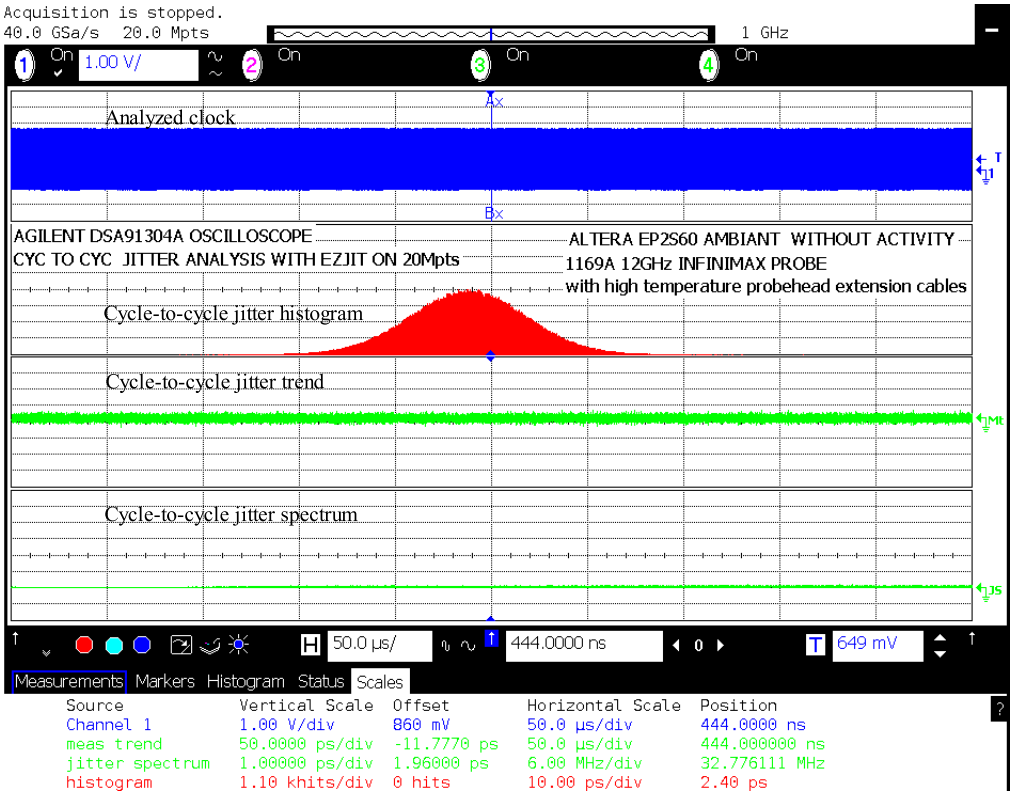

Figure 2-14 – Analyse de gigue cycle à cycle d'un RO implanté au sein d'un FPGA Altera EP2S60 porté à une température de 25°C.

La figure 2-15 montre qu'au sein du FPGA Xilinx, l'histogramme obtenu semble suivre une loi normale. De plus, comme le montre le spectre de puissance, la gigue cycle à cycle est

très proche d'un bruit blanc. Cependant, le spectre montre deux pics à 10.2 MHz et 20.4 MHz. L'amplitude de ceux-ci est inférieure à 1 ps mais révèle une partie déterministe au sein de la gigue cycle à cycle.

Figure 2-15 – Analyse de gigue cycle à cycle d'un RO implanté au sein d'un FPGA Xilinx V4SX35 porté à une température de 25°C.

3.1-2 Analyse de la gigue du RO soumis à une forte activité parasite

Les résultats de mesure obtenus lors de l'analyse de gigue d'un RO soumis à une forte activité parasite au sein d'un Altera Stratix II EP2S60 et d'un FPGA Xilinx Virtex-4 SX35 sont présentés respectivement dans les figures 2-16 et 2-17. Les histogrammes semblent suivre une loi normale. Comparée à la figure 2-14, la figure 2-16 montre qu'au sein du FPGA Altera, l'activité autour du RO engendre des pics d'amplitudes inférieures à 1 ps dans le spectre. Le même constat est effectué dans le FPGA Xilinx par l'analyse de la figure 2-17. Le spectre contient des parties déterministes non négligeables et relativement importantes. L'amplitude du pic maximale est égale à 4.255 ps et se situe à 36.23 MHz.

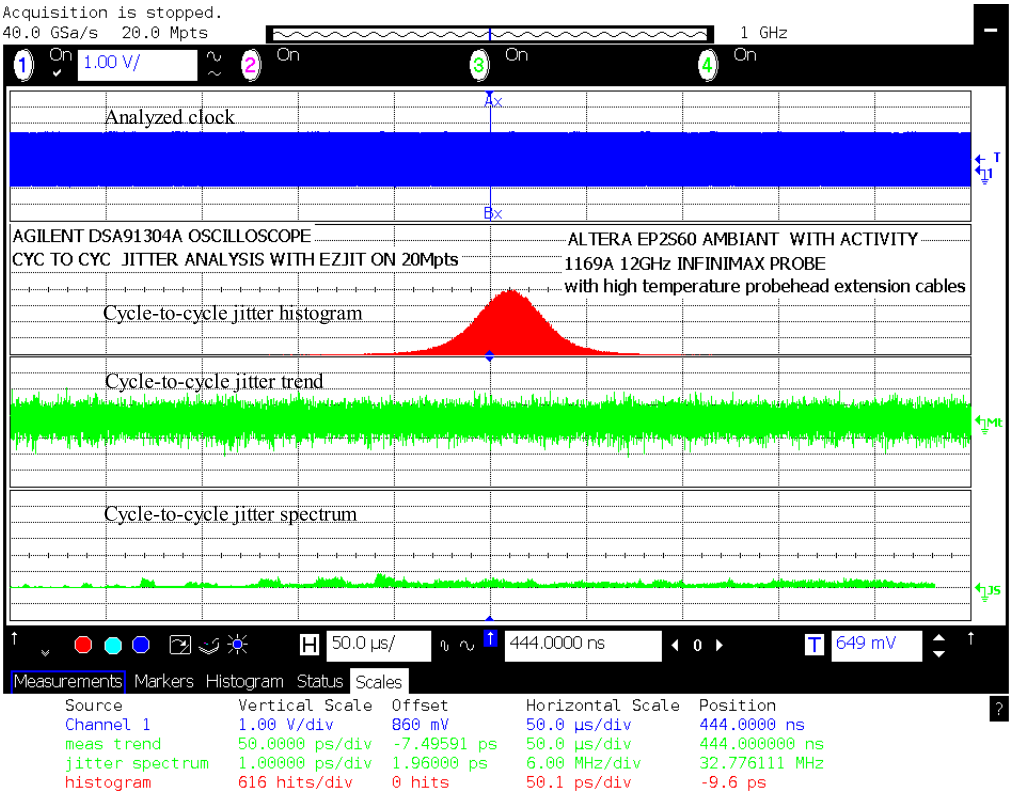

Figure 2-16 – Analyse de gigue cycle à cycle d'un RO soumis à une forte activité parasite implanté au sein d'un FPGA Altera EP2S60 porté à une température de 25°C.

3.1-3 Discussion

La variation de la température du FPGA a un impact négligeable sur les caractéristiques de la gigue cycle à cycle. Par contre, les mesures indiquent que la gigue varie en fonction du FPGA utilisé et de son activité. Lors d'une activité importante autour de l'oscillateur analysé, la partie déterministe est augmentée. Ceci laisse supposer que lorsque plusieurs RO sont routés de manière connexe, un couplage peut exister entre eux. Par conséquent, au sein d'un TRNG, l'activité des différents RO peut introduire de la corrélation entre les oscillateurs et réduire la qualité aléatoire du bruit échantillonné. De plus, les mesures montrent que l'oscillateur implanté au sein du FPGA Altera Stratix II est moins sensible à l'activité environnante qu'avec le FPGA Xilinx Virtex-4.

Il est donc nécessaire de modifier la méthodologie d'évaluation des TRNG sélectionnés. Le débit des générateurs peut être adapté en fonction du FPGA utilisé et de l'environnement de la source d'aléa. De plus, les propriétés statistiques du bruit échantillonné doivent être analysées. La partie suivante présente la méthode d'évaluation proposée.

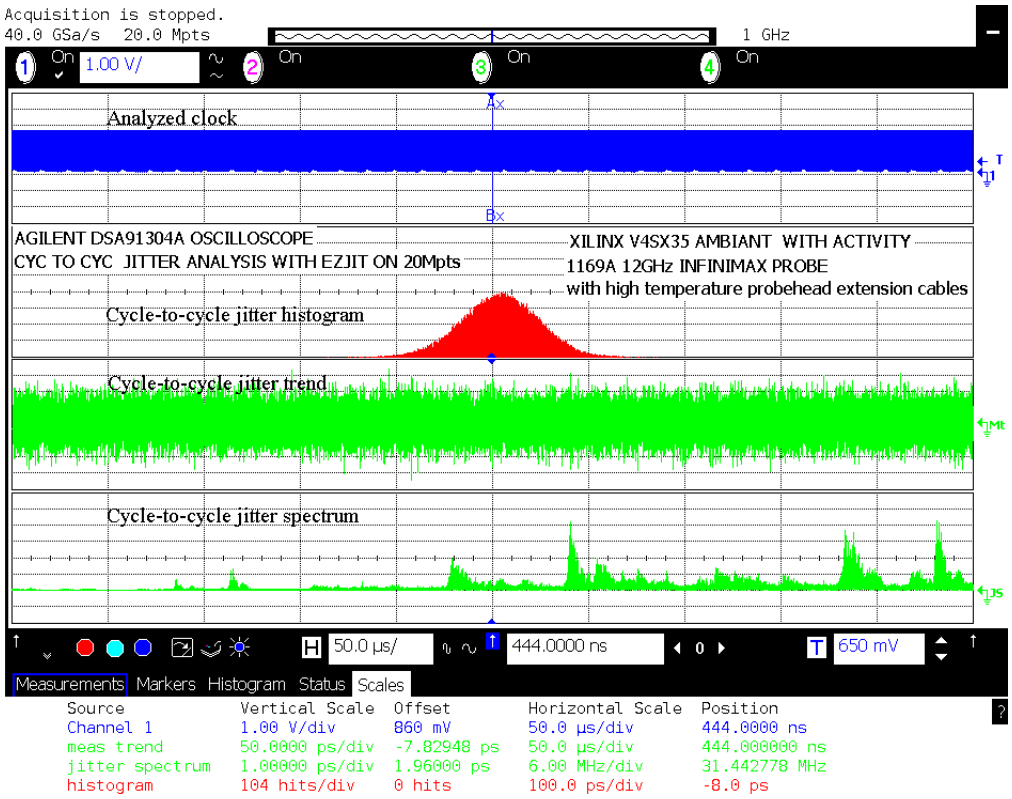

Figure 2-17 – Analyse de gigue cycle à cycle d'un RO soumis à une forte activité parasite implanté au sein d'un FPGA Xilinx V4SX35 porté à une température de 25°C.

4 Recherche du débit optimal des TRNG en fonction du FPGA ciblé

4.1 Objectif

Comme précédemment introduit, la gigue d'un signal oscillant varie d'un circuit à un autre et est fonction de l'activité du circuit. Implanter un générateur dans un certain circuit FPGA peut engendrer la modification de ses caractéristiques (débit, qualité). L'objectif de cette partie est de proposer une méthodologie permettant de trouver le débit optimal des générateurs [SMS07] et [DG07] en fonction du FPGA utilisé, un FPGA Xilinx Virtex-5 SX50. Le circuit est différent de ceux utilisés par [SMS07] et [DG07].

4.2 Méthodologie proposée

Afin d'évaluer les générateurs dans des conditions d'utilisation similaires, les TRNG sont implantés au sein du même circuit FPGA. Le principe de la méthode est proche de celle présentée dans la partie 2.2-6. Cependant, il est nécessaire de tester la qualité aléatoire de la séquence issue de l'échantillonnage de la source de bruit des générateurs. Une possibilité est d'utiliser des tests statistiques plus puissants tels que ceux proposés par l'AIS 31.

L'AIS 31 décompose les TRNG en deux classes, P1 et P2. D'après [AIS01], un générateur vraiment aléatoire est qualifié de P1 (P1-TRNG), si les nombres aléatoires générés après post-traitement valident les tests statistiques décrits dans [KS01]. Dans les applications nécessitant une sécurité accrue, génération de clé, *signature key pairs* ou *signature parameters* [SK03], les P1-TRNG ne sont pas suffisants. Une classe plus restrictive sur la qualité de la séquence aléatoire est introduite, la classe P2. Dans les P2-TRNG, la source de bruit échantillonnée (signal DAS pour *Digitized Analog Signal*) est aussi analysée. Par ailleurs, pour les deux classes, les TRNG sont définis par le niveau (bas, moyen ou élevé) de robustesse face à des attaques directes par exploitation des défaillances causées par leurs implantations [KS01]. Pour les niveaux moyen et élevé, la sortie des TRNG doit être validée en temps réel afin de désactiver la sortie du générateur en cas de défaillance.

Par ailleurs, la méthode d'évaluation doit permettre de trouver le débit optimal des générateurs. La solution proposée, présentée dans la figure 2-18, est d'utiliser un générateur d'horloge reconfigurable dynamiquement. Celle-ci repose sur l'utilisation des tests statistiques décrits par l'AIS 31 et ceux du FIPS 140-2. Les tests sont sélectionnés comme métrique afin d'évaluer la qualité des TRNG.

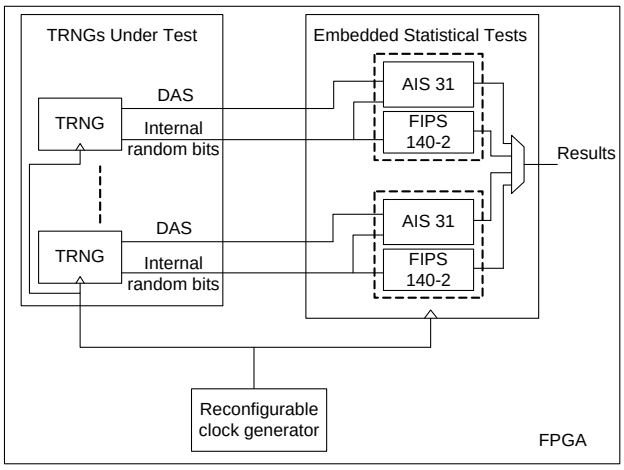

Figure 2-18 – La solution proposée pour optimiser le débit des TRNG est d'utiliser un générateur d'horloge reconfigurable dynamiquement. Celle-ci repose sur l'utilisation des tests statistiques décrits par l'AIS 31 et ceux du FIPS 140-2. Les tests sont sélectionnés comme métrique afin d'évaluer la qualité des TRNG.

4.3 IMPLANTATION MATÉRIELLE DE L'AIS 31

Cette section décrit l'implantation des tests statistiques de l'AIS 31 [KS01]. Afin de certifier un générateur de classe P2, l'AIS 31 implique la validation de neuf tests sta-

tistiques. Les tests T0-T5 sont exécutés à partir de la séquence aléatoire générée après post-traitement. Les tests T6-T8 sont appliqués à partir du bruit échantillonné DAS.

En raison du coût matériel du test T0, celui-ci ne peut pas être implanté dans le Xilinx Virtex 5 SX50 utilisé. Cependant, l'architecture du test est tout de même présentée dans la suite du rapport. Par ailleurs, les tests T1-T4 correspondent à la première version du FIPS 140-2, le FIPS 140-1. L'implantation des tests est très proche de celle étudiée dans la partie 2.2. Les tests utilisent des bornes différentes que celles détaillées dans la sous-section 2.2.

4.3-1 Test T0 : test de différence

Dans le test de différence (*disjointness test*), la séquence binaire DAS, échantillonnée à la fréquence f_{DAS}, est divisée en 2^{16} sous-séquences (ou blocs) disjointes de 48 bits $w_1, \ldots, w_{2^{16}} \in \{0,1\}^{48}$. La séquence analysée valide le test statistique si les blocs sont tous différents deux à deux. Ici, une mémoire accessible par contenu (CAM) n'est pas réalisable en raison de la taille mémoire nécessaire (2^{48} bits). La solution proposée est de trier les blocs dans l'ordre croissant. Une fois ceux-ci triés, un test analysant si deux blocs sont similaires est effectué. Durant la phase d'acquisition (réception de la séquence $w_1, \ldots, w_{2^{16}}$), un nouveau bloc est reçu tous les 48 cycles de l'horloge f_{DAS}. Comme le montre la figure 2-19, la solution est de trier la séquence binaire en série. A chaque réception d'un nouveau mot de 48 bits, celui-ci est trié.

Le principe de base du tri est d'utiliser 2^{15} bancs de deux registres 48 bits. Les bancs sont triés lors de la réception d'un nouveau mot. Comme le montre la figure 2-20, lors de la phase d'acquisition, à chaque réception d'un nouveau mot, les bancs déterminent la valeur maximale entre la valeur en entrée et celle des deux registres. La valeur maximale est transférée en sortie du banc. Une fois les 2^{16} mots reçus, la seconde phase de tri commence. Les bancs de registres fonctionnent dans le sens inverse. La direction des bus entre les bancs est inversée. Durant 2^{16} cycles, les bancs évaluent la valeur minimale entre le contenu des registres et l'entrée. Comme le montre la figure 2-21, la valeur minimale est ensuite transférée en entrée du banc suivant.

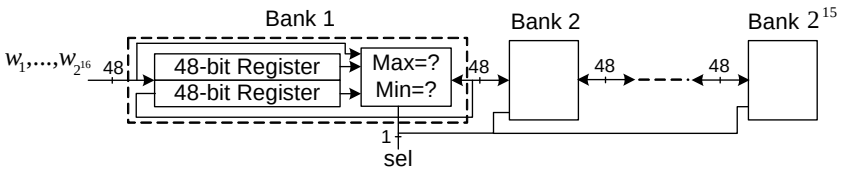

Figure 2-19 – Tri en série suivant l'ordre croissant des 2^{16} mots de 48 bits.

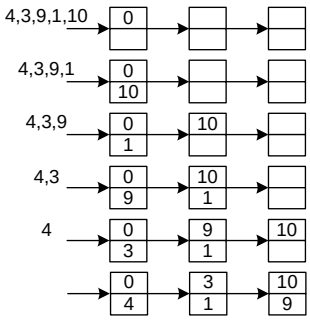

Figure 2-20 – Exemple de tri pendant la phase d'acquisition des 2^{16} mots de 48 bits. Lors de la phase d'acquisition, à chaque réception d'un nouveau mot, les bancs déterminent la valeur maximale entre la valeur en entrée et celle des deux registres.

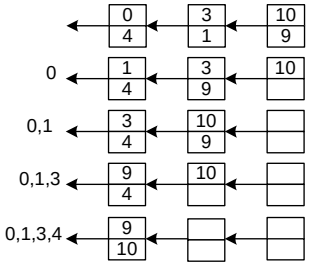

Figure 2-21 – Une fois les 2^{16} mots reçus, la seconde phase de tri commence. Les bancs de registres fonctionnent dans le sens inverse. La direction des bus entre les bancs est inversée. Durant 2^{16} cycles, les bancs évaluent la valeur minimale entre le contenu des registres et l'entrée. La valeur minimale est ensuite transférée en entrée du banc suivant.

Afin d'augmenter l'efficacité de la méthode de tri, l'architecture proposée à la figure 2-19 est modifiée. Instantier 2^{15} bancs composés de deux registres 48 bits est difficile en raison des ressources limitées. Dans une implantation FPGA, la solution à privilégier est d'utiliser les bancs mémoires disponibles dans le circuit. Par conséquent, la taille des bancs est augmentée. La contrainte à satisfaire est d'assurer le tri de chaque banc intermédiaire en 48 cycles d'horloge, soit $48 \times 1/f_{DAS}$ secondes. En pratique, la fréquence f_{DAS} est généralement faible, souvent quelques MHz (voir KHz). Par contre, la fréquence maximale des mémoires est généralement élevée. Une seconde horloge, de fréquence $f_{clk_{bram}}$, plus rapide, peut-être utilisée afin de synchroniser les blocs RAM. Soit Nb_{registre}, le nombre de registre 48 bits contenu dans un bloc RAM. La condition à satisfaire est

$$\frac{48}{f_{DAS}} \geq \frac{Nb_{\text{registre}} + 1}{f_{clk_{bram}}}. \tag{2-4}$$

La fréquence $f_{clk_{bram}}$ doit être choisie en fonction de la taille des blocs RAM, Nb_{registre} et de la fréquence f_{DAS}.

Cependant, l'implantation du test statistique oblige à utiliser une mémoire de taille

i	Intervalles requis pour n_i^0 et n_i^1
1	2267-2733
3	1079-1421
3	502-748
4	223-402
5	90-233
≥ 6	90-233

Tableau 2-5 – Intervalles requis dans le *run* test.

$2^{16} \times 48$ bits. Le coût en mémoire est trop important pour le FPGA ciblé. Une mémoire externe pourrait être utilisée. Cependant, l'utilisation de celle-ci obligerait à faire sortir le signal *das* à l'extérieur de FPGA. L'objectif final de la méthode d'évaluation est d'étudier la qualité des TRNG face aux attaques non-invasives. Des modifications pourraient être engendrées sur le signal *das*. Une solution basée sur l'utilisation d'une mémoire externe est écartée. Par conséquent, le test n'est pas implanté dans la méthode d'évaluation proposée.

4.3-2 Test T1 : *frequency* test

Le test vérifie l'uniformité d'une séquence aléatoire binaire constituée de 2×10^4 bits. Il s'agit du *frequency* test décrit dans la partie 2.2. Cependant, ici, le nombre de "1" doit appartenir à l'intervalle $[9655, 10345]$. Un compteur 14 bits est ici nécessaire.

4.3-3 Test T2 : *poker* test

Il s'agit du *poker* test décrit dans la partie 2.2. Cependant, le résultat de l'équation 2-2 doit appartenir à l'intervalle $]1.03; 57.4[$. Le coût de l'implantation est similaire à celle étudiée plus haut.

4.3-4 Test T3 : *run* test

Il s'agit du *run* test décrit dans la partie 2.2. Cependant, les nombres n_i^1 et n_i^0 doivent satisfaire les résultats présentés dans le tableau 2-5.

4.3-5 Test T4 : *long run* test

Il s'agit du *long run* test décrit dans la partie 2.2. Cependant, dans le test T4, aucun *run* ne doit dépasser une longueur de 34 ou plus.

4.3-6 TEST T5 : TEST D'AUTOCORRÉLATION

Dans le test d'autocorrélation, le test analyse une séquence aléatoire binaire composée de 2×10^4 bits. Les bits sont issus de la sortie du générateur après post-traitement $(b_1, \ldots, b_{2.10^4})$. Le TRNG valide le test si

$$Z_\tau = \sum_{j=1}^{5.10^3} (b_j \oplus b_{j+\tau}) \in]2326, 2674[, \text{ pour } \tau \in \{1, \ldots, 5.10^3\}, \tag{2-5}$$

où $\oplus$ désigne l'opération d'addition dans GF(2). Dans le test d'autocorrélation, un registre à décalage de 10.10^4 bits, un compteur 9 bits et un opérateur XOR un bit sont nécessaires. Il serait possible d'augmenter le nombre de XOR afin de diminuer la latence du test. Cependant, la latence du test T5 n'est pas la plus élevée de la batterie.

4.3-7 TEST T6 : TEST D'UNIFORMITÉ

Les P2-TRNG doivent satisfaire deux versions du test d'uniformité : les tests T6a et T6b.

TEST T6A

Dans le test T6a, une séquence binaire composée de 10^5 bits est analysée. Le test compte le nombre de "1" (n_1). Si n_1 appartient à l'intervalle requis $]47500, 52500[$, le TRNG passe avec succès le test. Un compteur 16 bits est seulement requis.

TEST T6B

Dans le test T6b, la séquence binaire analysée est décomposée en blocs indépendants de deux bits (w_{2j+1}, w_{2j+2}). Chaque bloc est utilisé pour générer deux ensembles $TF_{(0)}$ et $TF_{(1)}$. Le bloc (w_{2j+1}, w_{2j+2}) est affecté à l'ensemble $TF_{(r)}$ si $w_{2j+1} = r$. La séquence binaire est analysée tant que $TF_{(0)}$ et $TF_{(1)}$ ne contiennent pas au moins $N_1 = 10^5$ éléments. Les N_1 premiers blocs de $TF_{(r)}$ permettent de générer la valeur $St_{(r)}$. Lorsque $w_{2j+1} = r$, la valeur $St_{(r)}$, initialisée à 0 est incrémentée par w_{2j+2} jusqu'à ce que N_1 additions soient réalisées. Le nombre de "1" dans $St_{(0)}$ et $St_{(1)}$ est compté, produisant les valeurs $n_0(1)$ et $n_1(1)$.

Deux compteurs 17 bits sont nécessaires pour mémoriser le nombre d'éléments contenus dans $TF_{(r)}$ ainsi que les valeurs $n_r(1)$, pour $r = \{0, 1\}$. Ensuite, les valeurs $n_0(1)$ et $n_1(1)$ sont comparées. Si la condition

$$\left| \frac{n_0(1) - n_1(1)}{N_1} \right| < 2.10^{-2} \tag{2-6}$$

est satisfaite, le TRNG valide le test. En matériel, l'équation 2-6 est réalisée en testant si $|n_0(1) - n_1(1)| < 2.10^3$.

4.3-8 TEST T7 : EVALUATION DE LA DISTRIBUTION DES SOUS-SÉQUENCE BINAIRE

Les P2-TRNG doivent satisfaire deux versions du test T7 : les tests T7a et T7b.

TEST T7A

Dans le test T7a, la séquence binaire est décomposée en sous-blocs indépendants de 3 bits $(w_{3j+1}, w_{3j+2}, w_{3j+3})$. Le principe du test est proche du test T6a. La seule différence est d'analyser des blocs de 3 bits. Chaque bloc est utilisé pour générer quatre ensembles disjoints $TF_{(0,0)}, TF_{(0,1)}, TF_{(1,0)}$ et $TF_{(1,1)}$. Le bloc $(w_{3j+1}, w_{3j+2}, w_{3j+3})$ permet de construire l'ensemble $TF_{(r,s)}$ si $(w_{3j+1}, w_{3j+2}) = (r, s)$. La valeur $St_{(r,s)}$, initialisée à 0 est incrémentée par w_{3j+3} jusqu'à obtenir $N_1 = 10^5$ additions sur $St_{(r,s)}$. Le nombre de "1" (respectivement de "0") dans chaque ensemble $St_{(r,s)}$ est compté, produisant les valeurs $n_{r,s}(1)$ (respectivement $n_{r,s}(0)$). Ensuite, pour chaque $s \in \{0, 1\}$, les distributions $St_{(0,s)}$ et $St_{(1,s)}$ sont testées. Pour $r = \{0, 1\}$, si la condition

$$\frac{(n_{r,0}(0) - n_{r,1}(0))^2}{n_{r,0}(0) + n_{r,1}(0)} + \frac{(n_{r,0}(1) - n_{r,1}(1))^2}{n_{r,0}(1) + n_{r,1}(1)} < 15.13, \tag{2-7}$$

est satisfaite, le TRNG valide le test statistique.

En matériel, deux compteurs 17 bits sont nécessaires pour mémoriser le nombre d'éléments contenus dans $TF_{(r,s)}$ et la valeur de $n_{(r,s)}(1)$. Ensuite, l'équation 2-7 est réalisée en testant si la condition

$$\begin{aligned} 2N_1 \left(n_{r,0}(1) - n_{r,1}(1)\right)^2 \leq \quad & 15.13 \left(n_{r,0}(1) + n_{r,1}(1)\right) \\ & \times \left(2N_1 \left(n_{r,0}(1) + n_{r,1}(1)\right)\right) \end{aligned} \tag{2-8}$$

est validée. L'équation 2-8 repose sur le fait que $n_{r,k}(0) = N_1 - n_{r,k}(1)$, pour $k = \{0, 1\}$. Le coefficient 15.13 est codé en virgule fixe avec cinq bits de précision pour la partie fractionnaire. Le graphe flot de signal réalisant le calcul de l'équation 2-8 est présenté dans la figure 2-22.

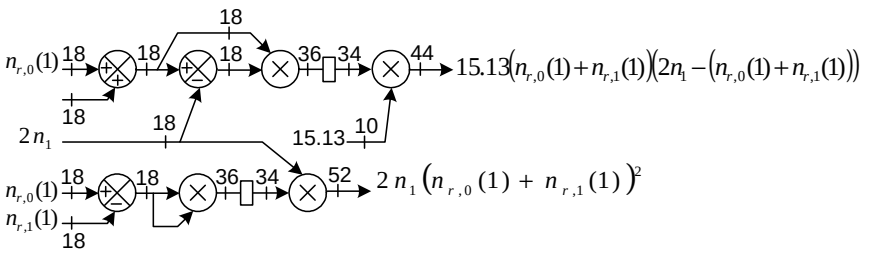

Figure 2-22 – Graphe flot de signal nécessaire à la réalisation du test T7a.

Test T7b

À la différence du test T7a, le test T7b décompose la séquence binaire analysée en blocs disjoints de quatre bits $(w_{4j+1}, w_{4j+2}, w_{4j+3}, w_{4j+4})$. Les blocs sont utilisés pour créer les huit sous-ensembles disjoints $TF_{(r,s,t)}$, où $r \in \{0,1\}$, $s \in \{0,1\}$ et $t \in \{0,1\}$. Ensuite, en utilisant le même fonctionnement que le test T7a, l'ensemble $St_{(r,s,t)}$ est construit à partir de N_1 additions de w_{4j+4} pour chaque ensemble $TF_{(r,s,t)}$. La suite du fonctionnement du test est similaire à celui du test T7a. Il en est de même de son implantation. La seule différence est le nombre de compteurs 17 bits requis, égal à 16.

4.3-9 Test T8 : test d'entropie

Principe du test d'entropie

Le test d'entropie est présenté par Jean-Sébastien Coron dans [Cor99] et [CN98]. Le test est basé sur le Test Universel de Maurer (TUM) [Mau91] et est inclus dans plusieurs batteries de tests statistiques (i.e. NIST). Le TUM est capable de détecter une grande partie des défaillances statistiques d'une séquence aléatoire. Le principe du test est de fournir une mesure asymptotique de l'entropie de la source. [CN98] propose une version modifiée du TUM produisant un calcul plus précis de l'entropie ainsi qu'une détection plus efficace des défaillances statistiques de la source testée.

Dans l'AIS 31, la séquence binaire analysée est décomposée en $Q+K$ séquences disjointes de L bits, $w_1, \ldots, w_{Q+K}$. Le test est basé sur le calcul de la distance minimale (A_n) entre w_n et le plus récent bloc similaire rencontré

$$A_n = \begin{cases} n \text{ si } \forall i < n, \text{ si } w_{n-i} \neq w_n \\ \min\{i : i \geq 1, w_n = w_{n-i}\} \text{ sinon.} \end{cases} \tag{2-9}$$

Le résultat de

$$f_{T_U}^H(s^N) = \frac{1}{K} \sum_{n=Q+1}^{K+Q} g(A_n), \tag{2-10}$$

où

$$g(i) = \frac{1}{log(2)} \sum_{k=1}^{i-1} \frac{1}{k}, \tag{2-11}$$

est calculé. En fonction des valeurs de L, Q et K, le résultat de (2-10) est comparé aux valeurs attendues. Dans l'AIS 31, le test utilise les grandeurs suivantes : $L = 8$, $Q = 2560$ et $K = 256.10^3$. D'après [KS01], si le résultat de (2-10) est supérieur à 7.976, le TRNG valide le test statistique.

L'implantation du test d'entropie peut être décomposée en deux parties. Dans la première partie, une mémoire de 256×18 bits est nécessaire pour sauvegarder le numéro du cycle (n) dans lequel le bloc w_n est rencontré. Ensuite, chaque fois qu'un nouveau mot est reçu, la distance A_n est calculée à partir du contenu de la mémoire. Le résultat $g(A_i)$ est calculé et additionné à la somme (2-10).

Pour diminuer la complexité arithmétique de (2-10), seul le calcul de

$$K \log(2) f_{T_U}^H(s^N) \tag{2-12}$$

est nécessaire. Cependant, si la complexité est réduite, une solution efficace à l'évaluation de la fonction $g(i)$ doit être trouvée. La fonction $g(i+1)$ correspond au nombre harmonique d'ordre i

$$H_i = \sum_{k=1}^{i} \frac{1}{k} \tag{2-13}$$

et doit être calculée pour $i \in \{1, \ldots, 258559\}$. La solution proposée pour implanter la fonction $g(i+1)$ est d'utiliser une approximation de H_i. Pour plus de détails concernant l'approximation de la fonction harmonique, se référer à [Vil07]. Dans ce travail, l'approximation de DeTemple-Wang [DW91]

$$H_i^{'} = \log\left(i + \frac{1}{2}\right) + \gamma + \frac{1}{24\left(i + \frac{1}{2}\right)^2 + \frac{21}{5}} \tag{2-14}$$

a été choisie, où γ désigne la constante d'Euler. Afin de simplifier l'écriture, γ est soustrait à (2-14) et est inclus dans le test final du résultat. La fonction à approchée est

$$H^{''}(i) = H^{'}(i) - \gamma = \log\left(i + \frac{1}{2}\right) + \frac{1}{24\left(i + \frac{1}{2}\right)^2 + \frac{21}{5}}. \tag{2-15}$$

Dans ce travail, l'erreur d'approximation (ϵ) est exprimée directement ou en terme de précision. La précision d'un calcul désigne le nombre de bits corrects donné par $\mu = -\log_2 |\epsilon|$.

Approximation polynomiale de $H^{''}(i)$

Une des solutions la plus répandue pour implanter une fonction f, avec une entrée et une sortie en virgule fixe (codées en complément à 2) est d'utiliser une approximation polynomiale. L'ensemble des polynômes de degré inférieur ou égal à d est noté $\mathcal{P}_d$. L'objectif est de trouver un polynôme $p^\star$ de $\mathcal{P}_d$ sur l'intervalle $[a, b]$ afin d'approcher au mieux la fonction f. Le polynôme $p^\star$ est calculé avec une précision plus importante que celle utilisée en pratique [Mul06]. Deux types d'approximation sont généralement considérées : l'approximation en moyenne quadratique, minimisant une erreur moyenne et l'approximation minimisant l'erreur au pire cas, appelée approximation minimax [Mul06]. Cette seconde méthode est adéquate afin de garantir une erreur maximale en sortie de l'approximation. Par conséquent, dans ce travail, celle-ci est sélectionnée afin de réaliser la recherche d'un polynôme idéal.

Il est nécessaire d'introduire la notion de distance, donnée par $\|f - p\|_\infty$

$$\|f - p\|_\infty = \max_{a \leq x \leq b} |f(x) - p(x)|. \tag{2-16}$$

Le polynôme $p^\star$ satisfait la relation

$$\|p^\star - f\|_\infty = \min_{p \in \mathcal{P}_n} \|p - f\|_\infty. \tag{2-17}$$

$p^\star$ est appelé le polynôme minimax de degré d approchant au mieux la fonction f sur l'intervalle $[a, b]$. Il est possible de trouver numériquement $p^\star$ grâce à l'algorithme de Remez [Rem34]. Cet algorithme utilise le théorème de Chebyshev montrant que $p^\star$ est le polynôme minimax de degré d de f sur $[a, b]$ si et seulement si il existe au moins $n + 2$ valeurs

$$a \leq x_0 < x_1 < \ldots < x_{d+1} \leq b \tag{2-18}$$

tel que

$$p^\star(x_i) - f(x_i) = (-1)^i \left[p^\star(x_0) - f(x_0)\right] = \pm\|f - p^\star\|_\infty. \tag{2-19}$$

En d'autres termes, le théorème montre que si $p^\star$ est le polynôme minimax de f, alors l'erreur maximale d'approximation est atteinte en au moins $d + 2$ points et que le signe de l'erreur oscille entre chaque point [Mul06]. Dans ce travail, le polynôme $p^\star$ est calculé numériquement en utilisant la fonction **minimax** de Maple$^{\text{TM}}$.

Soit $e^\star_{\text{app}}$, l'erreur d'approximation entre la fonction f et le polynôme minimax $p^\star$. En pratique, l'implantation du polynôme minimax engendre deux erreurs d'approximation supplémentaires, pouvant se cumuler à l'erreur d'approximation $e^\star_{\text{app}}$. La première erreur est causée par la quantification des coefficients de $p^\star$ en précision finie. La seconde erreur est provoquée par les bruits de calculs engendrés dans le graphe flot de signal (GFS) (on parle d'erreur d'arrondi).

Les coefficients du polynôme sont notés $p_0, \ldots, p_d$ et forment le polynôme $p(x) = \sum_{i=0}^{d} p_i x^i$. Les polynômes sont représentés en utilisant un codage en complément à 2. Plusieurs schémas peuvent être utilisés pour calculer le polynôme $p(x)$. Dans ce travail, le schéma de Horner

$$p(x) = p_0 + x\left(p_1 + x\left(p_2 + x\left(\ldots + xp_d\right)\ldots\right)\right) \tag{2-20}$$

est considéré. Celui-ci requiert d additions et d multiplications pour évaluer un polynôme de degré d. Il a aussi une erreur d'arrondi lors de l'évaluation du polynôme plus petite que le schéma directe.

L'erreur maximale en sortie de (2-10) doit être inférieure à 10^{-3} d'après [KS01]. Soit un nombre de bits corrects égal à 9.96, l'équation (2-10) est obtenue à partir de 256.10^6 additions successives de (2-15). L'erreur maximale en sortie de (2-15) est égale à $1/(256.10^6)$, soit une précision minimale de 27.963 bits corrects.

Dans un premier temps, l'objectif est de trouver le polynôme minimax en utilisant Maple$^{\text{TM}}$. L'entrée i de $H^{''}(i)$ est un entier 18 bits. Afin d'améliorer la vitesse d'approximation, l'entrée est normalisée entre $[2^{-18}, 258559 \times 2^{-18}]$. Par conséquent, la fonction à approcher est redéfinie par

$$H^{'''}(i) = \log\left(i \times 2^{18} + \frac{1}{2}\right) + \gamma + \frac{1}{24\left(i \times 2^{18} + \frac{1}{2}\right)^2 + \frac{21}{5}}. \tag{2-21}$$

La fonction $H'''(i)$ est présentée dans la figure 2-23. La fonction $H'''(i)$ est approchée pour $i \in [2^{-18}, 1[$.

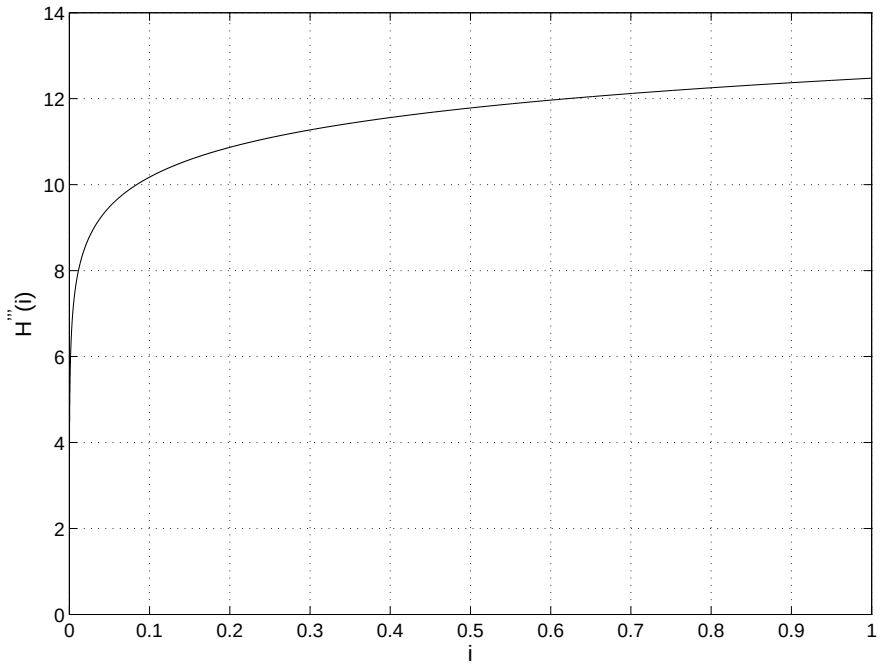

Figure 2-23 – Fonction $H'''(i)$ à approcher pour $i \in]0, 1[$. L'asymptote verticale pour les valeurs faibles de i rend l'approximation difficile.

Segmentation du domaine de définition de la fonction a implanter

Utiliser un seul polynôme pour approcher $H'''(i)$, avec $i \in [2^{-18}, 1[$ n'est pas possible en pratique (le degré du polynôme est trop élevé). Le domaine de la fonction doit être segmenté puis approché sur chacun d'eux. La solution la plus répandue est d'utiliser une segmentation uniforme. Le domaine de la fonction est divisé en segments de même taille. Si le problème d'adressage pour rechercher les coefficients des polynômes à utiliser est simple, les segments ne sont pas adaptés aux caractéristiques de la fonction. Par conséquent, selon la fonction, une segmentation uniforme peut être inefficace. Une autre possibilité est d'utiliser une segmentation non-uniforme dans laquelle la taille des segments peut varier.

Dans [LLVC03b], un schéma de segmentation est proposé : $\Delta_i = \{US, P2S\}$, dans lequel US et $P2S$ désignent respectivement une segmentation uniforme et par puissance de 2. Une segmentation hiérarchique à deux niveaux est réalisée $\Delta_0(\Delta_1)$ où Δ_0 et Δ_1 désignent respectivement le niveau 0 et 1. Les segments de niveau 0 utilisent une segmentation uniforme ou par puissance de 2. La segmentation de niveau 1 applique toujours une segmentation uniforme. Par conséquent, le schéma hiérarchique proposé est

$H = \{P2S(US), P2SL(US), P2SR(US), US(US)\}$. $P2SR$ (respectivement $P2SL$) désigne une segmentation $P2S$ diminuant vers le coté gauche (respectivement droit). Le nombre de bits requis pour adresser les segments de niveau 0 (v_0) est une contrainte du concepteur, au même titre que la fonction à approcher, la précision minimale et le schéma hiérarchique H. Le nombre de segments de niveau 1 nécessaire pour satisfaire la contrainte de précision est recherché. L'étape permet de trouver le nombre de bits (v_1) requis pour adresser les segments de niveau 1.

Dans cette étude, la segmentation non-uniforme proposée par Lee et al. dans [LLVC03b] a été sélectionnée. Comme le montre la figure 2-23, la fonction est difficile à approcher pour de petites valeurs de i. Par conséquent, dans ce travail, une segmentation de niveau 0 par puissances de 2 décroissantes seulement vers le coté droit (appelé P2SR dans [LLVC03b]) a été choisie. Le principe du P2SR est présenté dans le tableau 2-6 pour une entrée 8 bits, $v_0 = 5$ et $v_1 = 3$. Par conséquent, 5 et 8 segments de niveau 0 et 1 peuvent être adressés. Ici, une méthode différente de [LLVC03b] pour trouver v_1 est réalisée. Si la valeur de v_0 est petite, l'asymptote verticale de $H'''(i)$ nécessite un grand nombre de segments de niveau 1 ou des polynômes minimax de degrés élevés, soit un surcoût non négligeable en termes de ressources. Ici, la méthode proposée est de fixer le degré des polynômes minimax à $d = 3$, ainsi que le nombre de bits requis par la segmentation de niveau 0 : $v_0 = 18$ (la largeur de l'entrée). Le nombre de segments de niveau 1 pour satisfaire la précision minimale est recherché.

Pour chaque segment de niveau 0, l'algorithme est initialisé avec $v_1 = 0$ et recherche le polynôme minimax de degré 3. Si la précision obtenue est inférieure à la précision minimale (27.93 bits corrects), v_1 est incrémenté de 2. Le segment de niveau 1 est divisé en deux. Ensuite, le polynôme minimax est calculé et l'erreur d'approximation analysée. À chaque étape, le nombre de segment de niveau 1 est doublé. L'algorithme est répété tant que la précision obtenue est inférieure à la contrainte.

Chaque segment de niveau 0 est décomposé en 32 segments uniformes, d'où $v_1 = 5$. La figure 2-24 présente la segmentation de niveau 0 pour $i \in [2^{-18}, 1[$.

Ensuite, le nombre de bits minimal requis pour représenter les polynômes minimax en virgule fixe pour chaque segment de niveau 0 est recherché [MTVC06]. La taille de la partie fractionnaire est égal à $n = 28$ bits. La figure 2-25 présente la précision de l'approximation (en bits corrects) lorsque les polynômes sont codés en virgule fixe (avec $n = 28$ bits).

Approcher la fonction (2-21) pour $i \in [2^{-18}, 1[$ est inefficace. Comme précédemment introduit, v_0 a été fixé à 18, d'où l'utilisation de 18 segments de niveau 0 décomposés en 32 segments de niveau 1. En réalité, plusieurs segments de niveau 1 ne sont pas nécessaires.

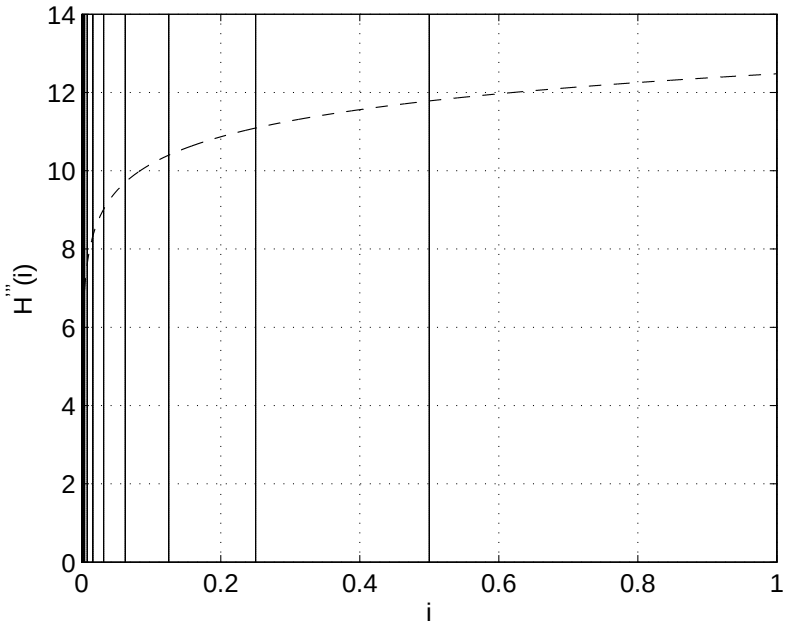

Figure 2-24 – Segmentation de niveau 0 pour $H'''(i)$ où $i \in [2^{-18}, 1[$.

adresses P2SR	Intervalle
0	**00000**|000 $\Rightarrow$ **00000**|111
1	**00001**|000 $\Rightarrow$ **0000.**|111
2	**00010**|000 $\Rightarrow$ **0001.**|111
3	**00100**|000 $\Rightarrow$ **001..**|111
4	**01000**|000 $\Rightarrow$ **01...**|111
5	**10000**|000 $\Rightarrow$ **1....**|111

Tableau 2-6 – Segmentation P2SR pour une entrée 8 bits, $v_0 = 5$ et $v1 = 3$.

Par exemple, le segment de niveau 1 $[2^{-18}, 2^{-17}[$ est aussi divisé en 32 segments uniformes. Dans ce cas de figure, l'approximation minimax est plus coûteuse qu'une mémorisation directe de (2-21) pour des valeurs petites de i. Ceci est dû au fait que la recherche du polynôme minimax est réalisée dans $\mathbb{R}$ sans prendre en compte un pas de quantification égal à 2^{-18}. Afin d'évaluer à partir de quel segment de niveau 0 l'approximation est intéressante, l'efficacité de l'approximation est évaluée en fonction du coût mémoire. La figure 2-26 montre la taille mémoire requise (en bits) en fonction de la méthode utilisée pour approcher la fonction $H'''(i)$ pour $i \in [2^{-18}, 2^{-18+j}[$ (approximation polynomiale ou mémorisation directe). Pour $i \in [2^{-18}, 2^{-13}[$, la mémorisation directe est plus efficace. Ensuite, à partir de $j = 6$, pour $i \in [2^{-13}, 1[$, l'approximation polynomiale est largement moins coûteuse en termes d'unité mémoire. Par conséquent, pour $i \in [2^{-18}; 2^{-13}[$, la précision obtenue en sortie de (2-21) est égale à 28 bits corrects.

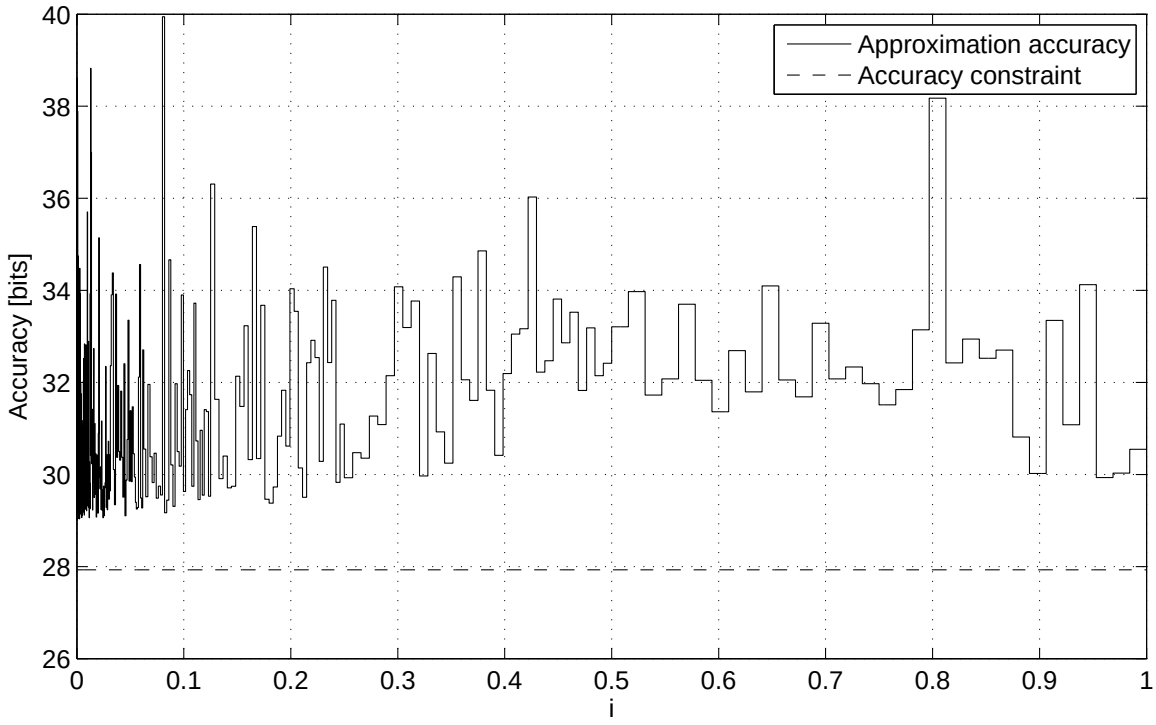

Figure 2-25 – Précision de l'approximation (en bits corrects) pour $i \in [2^{-18}, 1[$.

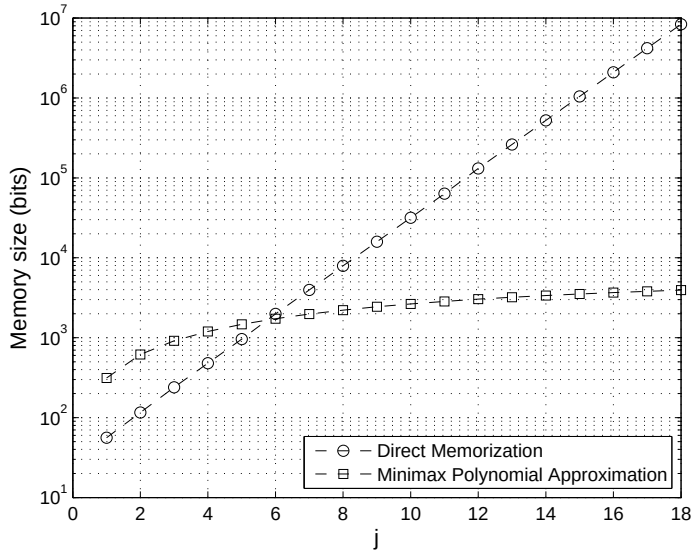

Figure 2-26 – Taille mémoire en fonction de la méthode utilisée pour approcher la fonction $H'''(i)$ pour $i \in [2^{-18}, 2^{-18+j}[$.

Ensuite, la taille minimale du chemin de données est calculée. Lors du calcul du GFS,

la réalisation d'une série d'additions suivie de troncatures engendrent un bruit de calcul non négligeable. La précision en est fortement dégradée. Afin de réduire le bruit de calcul, un certain nombre de bits de garde (g) sont ajoutés lors des calculs intermédiaires. Ici le Schéma de Horner

$$p^{\star}(x)_{\text{quantifié}} = ((p_3 \times x + p_2) \times x + p_1) \times x + p_0 \tag{2-22}$$

est utilisé. Pour chaque segment de niveau 1, la précision du schéma de Horner est calculée par des simulations en virgule fixe réalisées par Matlab. La figure 2-27 présente le GFS utilisé pour approcher la fonction (2-21) pour $i \in [2^{-13}, 1[$. Le bloc T désigne une opération de troncature réalisée sur la partie fractionnaire du chemin de données. Après chaque multiplication, le nombre de bits minimal requis par les parties fractionnaires est recherché. Une recherche exhaustive est réalisée. Le chemin de données doit contenir 30 bits pour la partie fractionnaire, soit 2 bits de garde. Dans la figure 2-27, les données sont représentées en virgule fixe. Celles-ci sont composées d'un bit de signe, d'une partie entière et d'une partie fractionnaire. Les données en virgule fixe sont spécifiées suivant la notation (b, m, n), où b, m et n désignent respectivement la largeur en bits de la donnée, de la partie entière et de la partie fractionnaire.

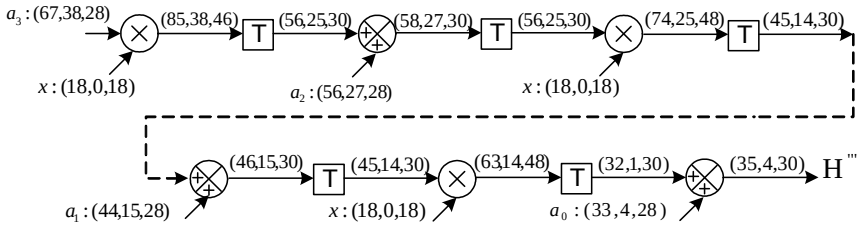

Figure 2-27 – Graph flot signal réalisant l'approximation de $H'''(i)$ pour $i \in [2^{-13}, 1[$.

La précision de l'approximation de $H'''(i)$ pour $i \in [2^{-13}, 2^{1}[$ en utilisant le GFS est présentée dans la figure 2-28. La précision minimale est égale à 27.94 bits corrects, soit une précision supérieure à la contrainte de précision égale à 27.93.

Architecture matérielle de l'approximation de $H'''(i)$

Un problème à étudier est celui de l'adressage des coefficients des polynômes. L'entrée i de $H'''(i)$ est un entier 18 bits. Les 13 bits de poids forts de i $((i_{17}, ..., i_5)_2)$ sont utilisés pour sélectionner le segment de niveau 0. Le zéro situé le plus à droite des 13 bits est détecté à partir d'un *Leading Zero Detector* (LZD). La solution proposée par [LLVC03b] et présentée dans la figure 2-29 a été choisie. Des opérateurs OU logiques appliqués en cascade sur les 13 bits de poids forts et un compteur de 1 sont requis. Le LZD produit la

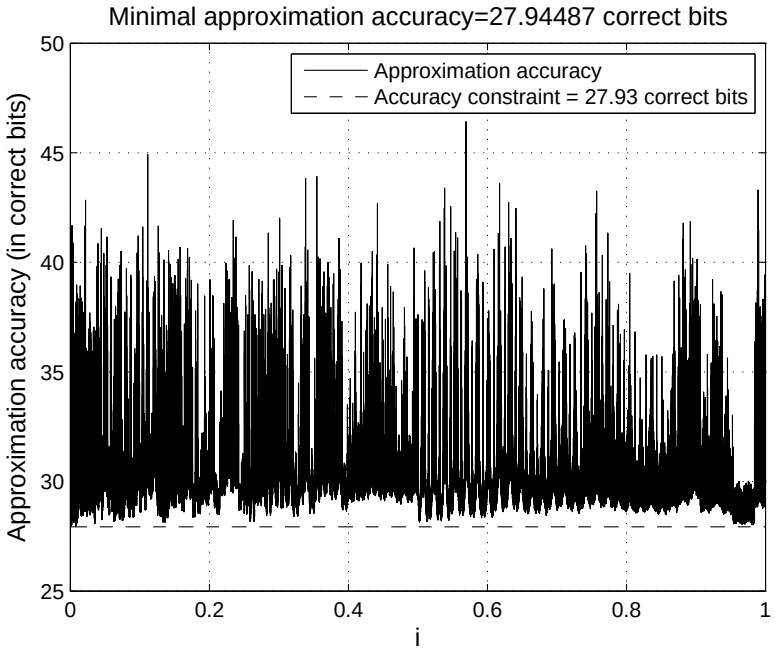

Figure 2-28 – Précision de l'approximation (en bits corrects) de $H'''(i)$ pour $i \in [2^{-13}, 1[$.

variable *P2SR address* pouvant prendre une valeur entre 1 et 13. Une fois le segment de niveau 0 sélectionné, le segment de niveau 1 est adressé en utilisant les 5 MSB suivant le $(\text{P2SR address} + 4)^{\text{ième}}$ bit de i.

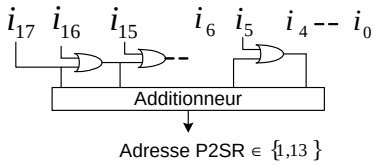

Figure 2-29 – *Leading Zero Detector* (LZD). Des opérateurs OU logiques appliqués en cascade sur les 13 bits de poids forts et un compteur de 1 sont requis. Le LZD produit la variable *P2SR address* pouvant prendre une valeur entre 1 et 13.

Les coefficients p_3, p_2, p_1 et p_0 sont mémorisés par quatre mémoires ROM de profondeur égale à 416. La figure 2-30 présente l'organisation dans la mémoire pour le coefficient p_0. Le coefficient à adresser est facilement calculé par

$$\text{addresse mémoire } = 2^5 \times (\text{adresse P2SR} - 1) + \text{position segment de niveau 1.} \quad (2\text{-}23)$$

Pour $i \in [2^{-18}, 2^{-13}[$, les 31 résultats de $H'''(i)$ sont mémorisés par une CAM dont la taille est égale à 31×31 bits.

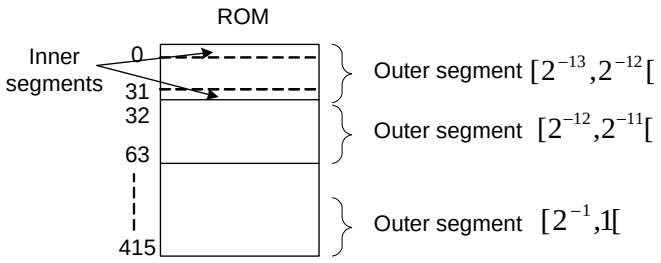

Figure 2-30 – Organisation de la ROM pour la mémorisation du coefficient p_0.

FPGA ciblé	Xilinx Virtex 5 SX50
Nombre total de LUT	499 (1%)
Fréquence maximale	84.2 MHz
BRAM	7 (5%)
DSP48	32 (11%)
Latence au pire cas	8 cycles

Tableau 2-7 – Résultats d'implantation pour une cible FPGA Xilinx Virtex 5 XC5VSX50.

Le circuit approximant la fonction $H'''(i)$ pour $i \in [2^{-18}, 1[$ est placé et routé par l'outil ISE 10.1 pour la cible FPGA Xilinx Virtex 5 SX50-1. Le tableau 2-7 résume les performances obtenues. Dans le pire cas, la latence du calcul est de 8 cycles.

4.3-10 PERFORMANCE ET COÛT MATÉRIEL DES TESTS DE L'AIS 31

Les tests statistiques de l'AIS 31 sont implantés dans le FPGA Xilinx Virtex 5 SX50-1. Les performances et coûts des tests et de la batterie complète de l'AIS 31 sont résumés dans le tableau 2-8. Ceux-ci ont été obtenus par l'outil Xilinx ISE 10.1.03. Comme indiqué, l'AIS 31 occupe une faible surface du FPGA ciblé (10 %). Malgré la taille relativement faible de la matrice du FPGA, l'implantation de l'AIS 31 est peu coûteuse. La fréquence maximale de 88.2 MHz autorise l'évaluation en ligne et temps réel de la qualité des TRNG.

4.4 EVALUATION DES GÉNÉRATEURS SÉLECTIONNÉS

Le schéma présenté en figure 2-18 est réalisé. Les générateurs sont implantés au sein du même FPGA, un Xilinx Virtex-5 SX50. Afin d'évaluer la qualité des TRNG en fonction de leur débit, une horloge reconfigurable réalisée dans le FPGA Xilinx par un bloc *Digital Clock Manager* (DCM) [Xil08] est utilisée. Pendant la phase de reconfiguration, l'évaluation des générateurs est stoppée. Une fois le re-verrouillage de la DCM assuré, les TRNG

Tests		
T1	Nombre total de LUTs	32 (0%)
	Fréq. Max	345.542 MHz
T2	Nombre total de LUTs	178 (0%)
	Fréq. Max	125.73 MHz
T3	Nombre total de LUTs	244 (0%)
	Fréq. Max	271.22 MHz
T4	Nombre total de LUTs	42 (0%)
	Fréq. Max	287.27 MHz
T5	Nombre total de LUTs	189 (0%)
	BRAM	1 (0%)
	Fréq. Max	288.6 MHz
T6a	Nombre total de LUTs	81 (0%)
	Fréq. Max	314.56 MHz
T6b	Nombre total de LUTs	245 (0%)
	Fréq. Max	221 MHz
T7a	Nombre total de LUTs	919 (2%)
	Fréq. Max	90.67 MHz
T7b	Nombre total de LUTs	1620 (4%)
	Fréq. Max	90.58 MHz
T8	Nombre total de LUTs	499 (1%)
	Fréq. Max	84.2 MHz
	BRAM	7 (5%)
	DSP48	32 (11%)
Total	Nombre total de LUTs	4042 (10%)
	Fréq. Max	84.2 MHz
	BRAM	8 (6%)
	DSP48	32 (11%)

Tableau 2-8 – Résultat de l'implantation des tests de l'AIS 31 pour une cible FPGA Xilinx Virtex 5 SX50T-1.

fonctionnent librement pendant 5.10^3 cycles d'horloge avant de procéder à l'évaluation par les tests statistiques. Les générateurs sont analysés pour des débits variant de 1.03 Mb/s à 10.65 Mb/s avec des pas de 343.7 Kb/s.

La qualité des générateurs est évaluée en deux étapes. La première étudie les propriétés de la séquence binaire après post-traitement par les tests *frequency*, T1, *poker*, T2, *run*, T3, *long-run*, T4 et T5. L'ensemble des tests est appelé T_{r}. Ensuite, la qualité aléatoire du bruit échantillonné est examinée par les tests T6a, T6b, T7a, T7b et T8. L'ensemble est désigné par T_{w}. Afin de faciliter l'analyse des générateurs, les taux d'échec obtenus par les TRNG aux ensembles T_{r} et T_{w} sont étudiés. Les taux sont divisés en quatre classes : la classe A pour un pourcentage d'échec de 0%, la classe B entre 0% et 10%, la classe C entre 10% et 30% et enfin, la classe D pour des taux d'échec supérieurs à 30%.

Les figures 2-31 et 2-32 correspondent aux pourcentages des quatre classes lorsque les générateurs [SMS07] et [DG07] sont analysés par l'ensemble de tests T_{r} pour différents débits de génération allant de 1.03 Mb/s à 10.65 Mb/s avec des pas de 343.7 Kb/s. Comme le montre la figure 2-31, le débit maximal du TRNG proposé par Sunar et al. [SMS07] est de 6.53 Mb/s. Pour des débits supérieurs à 7.56 Mb/s, la qualité du générateur est dégradée.

La figure 2-32 présente l'évaluation du TRNG présenté par Dichtl et al. [DG07]. D'après les tests T_{r}, le débit maximal du générateur est égal à 5.5 Mb/s pour le FPGA Xilinx Virtex 5 SX50 testé. Au delà de ce débit, la qualité du générateur est dégradée.

Les figures 2-33 et 2-34 présentent les résultats de l'évaluation des TRNG [SMS07] et [DG07] par les tests T_{w} pour différents débits de génération allant de 1.03 Mb/s à 10.65Mb/s avec des pas de 343.7 Kb/s. La figure 2-33 montre que le bruit échantillonné du TRNG proposé par Sunar et al. [SMS07] peut être utilisé jusqu'à un débit de 8.59 Mb/s. La figure 2-34 indique que le bruit échantillonné par le TRNG Dichtl et al. [DG07] peut être utilisé à un débit maximal de 6.53 Mb/s.

4.5 Conclusions de l'évaluation

Les expérimentations montrent que la diminution du débit des TRNG n'engendre pas forcément une amélioration des propriétés statistiques de la séquence aléatoire. Par exemple, lorsque le générateur Sunar et al. [SMS07] fonctionne à un débit inférieur au débit maximal, pour certains débits le générateur échoue aux tests statistiques. Nous supposons que

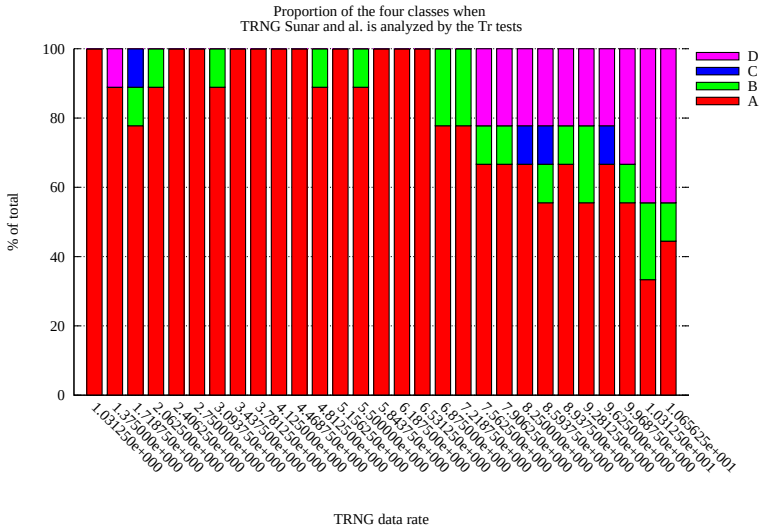

Figure 2-31 – Pourcentage des quatre classes lorsque le TRNG Sunar et al. [SMS07] est analysé par l'ensemble des tests T_r tests pour différents débits de génération allant de 1.03 Mb/s à 10.65Mb/s avec des pas de 343.7 Kb/s. L'axe des abscisses est ici exprimé en Mb/s.

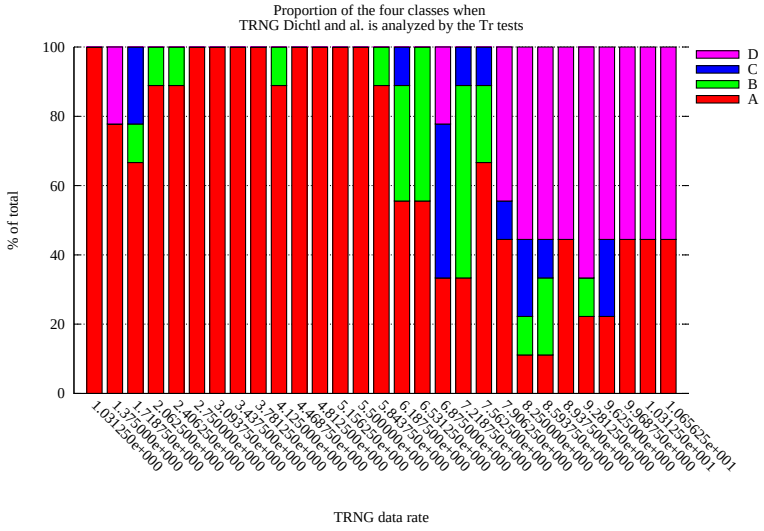

Figure 2-32 – Pourcentage des quatre classes lorsque le TRNG Dichtl et al. [DG07] est analysé par l'ensemble des tests T_r tests pour différents débits de génération allant de 1.03 Mb/s à 10.65Mb/s avec des pas de 343.7 Kb/s. L'axe des abscisses est ici exprimé en Mb/s.

ceci est provoqué par le fait que la fréquence d'échantillonnage du TRNG est un multiple commun de la fréquence du bruit échantillonné.

L'analyse montre que le TRNG Sunar et al. [SMS07] semble être plus robuste que le générateur Dichtl et al. [DG07]. Par ailleurs, le débit de [SMS07] est plus élevé.

De plus, les résultats montrent que le débit maximal du signal DAS produit par les deux générateurs est supérieur à celui atteignable par la sortie des TRNG. Par exemple, le

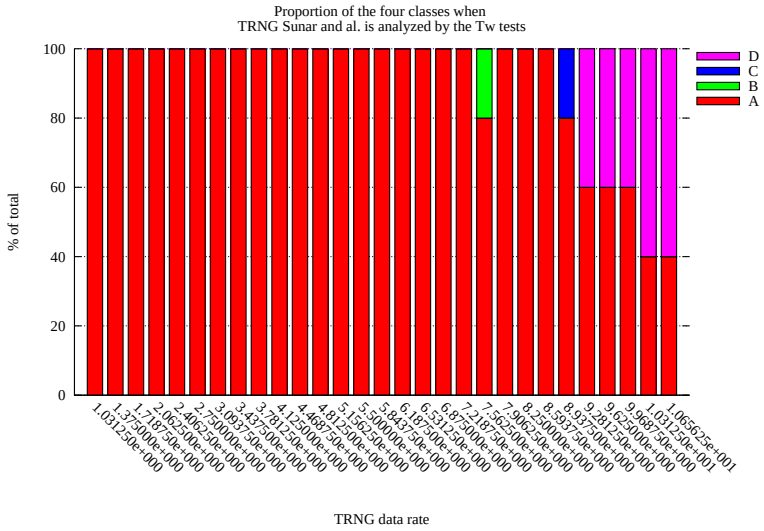

Figure 2-33 – Pourcentage des quatre classes lorsque le TRNG Sunar et al. [SMS07] est analysé par l'ensemble des tests T_{w} tests pour différents débits de génération allant de 1.03 Mb/s à 10.65Mb/s avec des pas de 343.7 Kb/s. L'axe des abscisses est ici exprimé en Mb/s.

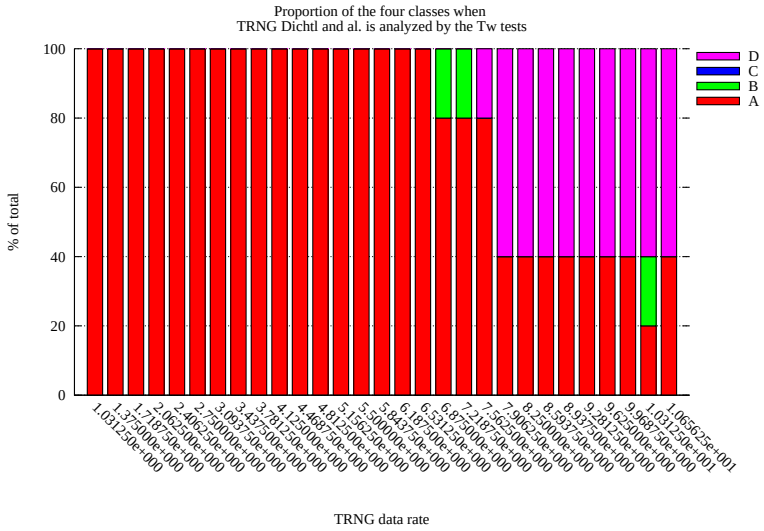

Figure 2-34 – Pourcentage des quatre classes lorsque le TRNG Dichtl et al. [DG07] est analysé par l'ensemble des tests T_{w} pour différents débits de génération allant de 1.03 Mb/s à 10.65Mb/s avec des pas de 343.7 Kb/s. L'axe des abscisses est ici exprimé en Mb/s.

débit maximal du TRNG proposé par Sunar et al. [SMS07] est de 8.59 Mb/s. En sortie du bloc de post-traitement, le débit maximal est de 6.53 Mb/s. Le même constat est réalisé sur le TRNG présenté par Dichtl et al. [DG07]. Ceci montre que des optimisations sur les blocs de post-traitement des TRNG peuvent améliorer le débit des générateurs.

Enfin, la méthode proposée permet d'adapter le débit des générateurs en fonction du

TRNG	Nombre ALUT
Sunar and al. [SMS07]	1 652
Vasyltsov and al. [VHKK08]	9
Dichtl and al. [DG07]	73

Tableau 2-9 – Nombre d'ALUT requis par les TRNG au sein du FPGA Altera EP2S60.

circuit utilisé et de son activité. Par exemple, au sein d'un FPGA Xilinx Virtex 2, le débit obtenu par [SK03] (implantation de [SMS07]) est égal à 2.5 Mb/s. Pour ce même générateur dans le FPGA Xilinx Virtex 5 SX50 testé, la méthode proposée permet d'obtenir un débit maximal de 6.53 Mb/s pour la séquence après post-traitement et de 8.59 Mb/s pour le bruit échantillonné. [DG07] spécifie un débit de 6.25 Mb/s au sein d'un FPGA Altera Spartan-3. Ici, la méthode converge vers un débit de 5.5 Mb/s pour la séquence après post-traitement et de 6.53 Mb/s pour le bruit échantillonné.

5 Evaluation du coût matériel et de la puissance consommée des TRNG testés

5.1 Coût matériel

Chaque TRNG est implanté dans une carte de développement FPGA Altera Stratix II EP2S60. La surface des générateurs est reportée dans le tableau 2-9 en fonction du nombre total d'ALUT retourné par le logiciel Altera Quartus 7.2. Le TRNG le moins coûteux matériellement est le TRNG [VHKK08]. Le générateur consomme seulement quelques ALUT. Le générateur [DG07] nécessite légèrement plus de ressource. À la différence des deux précédents générateurs, le TRNG [SMS07] requiert un grand nombre d'ALUT. Par conséquent, d'un point de vue matériel, les TRNG sont à privilégier en vue d'une implantation au sein d'un circuit cryptographique à faible coût.

5.2 Puissance consommée

Estimer la puissance consommée en utilisant les outils de développement des constructeurs de FPGA donne des résultats peu crédibles. Les outils semblent avoir des difficultés pour simuler la métastabilité des signaux rencontrés dans les générateurs de nombres vraiment aléatoires.

L'alternative privilégiée est de mesurer directement la puissance consommée par le cœur du FPGA. La carte de développement FPGA Actel Igloo permet de mesurer le courant consommé par le cœur du FPGA. Par conséquent, il est possible de calculer la puissance dynamique consommée. Les résultats obtenus pour les générateurs sont résumés dans le

TRNG	Puissance dynamique (mW)
Sunar and al. [SMS07]	45.36
Vasyltsov and al. [VHKK08]	2.52
Dichtl and al. [DG07]	3.96

Tableau 2-10 – Puissance dynamique consommée par les trois générateurs implantés au sein de la carte de développement FPGA Actel Igloo.

tableau 2-10. La consommation des générateurs est relativement faible. De plus, cet effet est amplifié par le fait d'utiliser un FPGA très faible consommation. Cependant, en fonctionnement, la puissance consommée par le TRNG [SMS07] est supérieure d'un facteur 18 et 11.5 par rapport à [VHKK08] et [DG07]. Le tableau montre aussi que la puissance consommée par [VHKK08] and [DG07] est proche. De plus, la très faible consommation du générateur [DG07] pendant l'étape de repos montre son intérêt pour les applications nécessitant seulement occasionnellement un nouveau vecteur d'initialisation.

6 Conclusions

Après avoir présenté les trois TRNG sélectionnés, la robustesse des générateurs au sein d'un FPGA Altera Stratix II est évaluée. L'objectif de l'étude était d'étudier la qualité des générateurs en fonction de la température du circuit FPGA et de son activité. Une méthodologie d'évaluation, basée sur l'utilisation des tests statistiques du FIPS 140-2 est proposée. À partir de l'implantation matérielle des tests, les générateurs sont évalués, en temps réel et au sein du même circuit. La méthode permet de comparer le comportement des générateurs lorsque ceux-ci sont utilisés dans des environnements similaires. Cette étude a permis de retenir deux générateurs, le TRNG de Sunar et al. [SMS07] et celui de Dichtl et al. [DG07].

Les deux générateurs sont basés sur la gigue d'un ou plusieurs oscillateur(s). À partir de mesures réelles de gigue réalisées, l'importance d'adapter le comportement des TRNG en fonction du FPGA utilisé et de son activité est présentée. Par conséquent, la méthodologie d'évaluation des TRNG a été modifiée afin de chercher le débit maximal des générateurs. La méthode présentée teste à la fois la séquence binaire en sortie du générateur ainsi que les propriétés statistiques du signal DAS. Celle-ci utilise une implantation efficace des tests statistiques proposés par l'AIS 31. Cette étude a permis d'analyser les performances des générateurs de Sunar et al. [SMS07] et de Dichtl et al. [DG07] au sein d'un FPGA Xilinx Virtex 5 SX50. La robustesse du générateur proposé par Sunar et al. a été montrée. Cependant, le coût matériel et la consommation engendrés par ce générateur sont relativement importants. Par conséquent, le TRNG proposé par Dichtl et al. sera préféré au sein des circuits présentant de fortes contraintes en surface et en consommation. De plus, la méthode proposée a permis d'optimiser les débits des TRNG au sein du FPGA visé.

Cette étude a montré la possible variation de la qualité d'un TRNG en fonction de l'environnement extérieur et de l'activité du circuit dans lequel celui-ci est implanté. Par conséquent, des mécanismes de test en ligne permettant d'analyser, en temps réel, la qualité d'un générateur doivent être mis en œuvre. La figure 2-35 présente un RNG hybride disposant d'un tel mécanisme et proposé dans cette thèse. À partir de l'évaluation de la qualité du signal DAS et de la séquence binaire produite après post-traitement, le comportement du TRNG est analysé. En cas de défaillances importantes du TRNG, une alarme est levée. L'analyse, en temps réel de la qualité du TRNG permet d'assurer un excellent germe au PRNG. Cependant, pour être valables, les tests en ligne doivent être peu coûteux en termes de ressources matérielles. Le chapitre suivant décrit l'implantation d'un tel système.

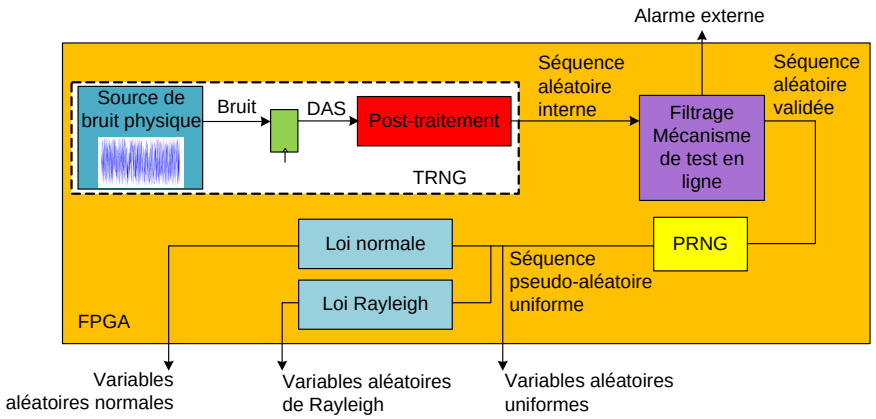

Figure 2-35 – Présentation d'un RNG hybride disposant de mécanisme de test en ligne de la qualité du TRNG. À partir de l'évaluation de la qualité du signal DAS et de la séquence binaire produite après post-traitement, le comportement du TRNG est analysé. En cas de défaillances importantes du TRNG, une alarme est levée. L'analyse, en temps réel de la qualité du TRNG permet d'assurer un excellent germe au PRNG.

Chapitre

3 Vers des RNG uniformes et gaussiens à très haut débit

Comme l'a présenté le chapitre II, un générateur hybride doit être constitué de mécanismes de test en ligne permettant la validation, temps réel, de la qualité d'un TRNG. La première section de ce chapitre décrit le développement d'un RNG hybride conçu conjointement avec une entreprise spécialisée en cryptographie. L'entreprise propose une solution cryptographique nécessitant l'implantation au sein d'un circuit reconfigurable d'un RNG à très haut débit. La section décrit le circuit désiré par l'entreprise.

Dans certaines applications, une implantation au sein de circuits dédiés (e.g. ASIC) est souvent requise. Si le chapitre II a permis de sélectionner un TRNG au sein d'une plateforme reconfigurable, le générateur à choisir peut être différent au sein d'une solution ASIC. La seconde section du chapitre présente l'implantation de deux RNG hybrides au sein de circuits ASIC. Les circuits permettront la sélection d'un TRNG optimal en vue d'une implantation au sein d'un circuit dédié.

Les générateurs hybrides permettant de générer des nombres aléatoires uniformes sont très utilisés dans les applications cryptographiques. Cependant, dans certaines applications, il est parfois nécessaire de générer des nombres aléatoires suivant une distribution de probabilité différente. En communication numérique, il est souvent nécessaire de connaître le taux d'erreur binaire d'un système. Ainsi, des modèles de canaux de transmission sont souvent utilisés pour simuler le comportement de tels systèmes. Cependant, les taux d'erreur binaire actuels (10^{-9}) entraînent la réalisation d'un grand nombre de simulations. La complexité des modèles de canaux de transmission engendre des temps de calcul très importants. Une alternative afin d'améliorer la rapidité des simulations est d'implanter le modèle de canal de transmission en matériel, au sein de circuits reconfigurables. La dernière section du chapitre présente la conception matérielle du circuit permettant la simulation temps réel d'un canal à bruit gaussien additif et d'un canal de Rayleigh.

1 Implantation d'un RNG hybride au sein d'un circuit reconfigurable

L'objectif de cette section est de présenter l'implantation FPGA d'un RNG hybride réalisé en collaboration avec une entreprise spécialisée en cryptographie. L'entreprise a développé une solution permettant d'assurer la confidentialité, l'intégrité et/ou la non interception des données transmises.

1.1 Description du générateur hybride

L'architecture du générateur hybride a été conçue en collaboration avec l'entreprise spécialisée en cryptographie. Au sein de la solution, notre rôle a été de concevoir et de développer le TRNG. De plus, un mécanisme de contrôle de la qualité du générateur devait être réalisé afin d'assurer la qualité du TRNG. Par ailleurs, cette contrainte est imposée par la DCSSI (Direction centrale de la sécurité des systèmes d'Information).

L'entreprise désirait obtenir un générateur facilement implantable au sein d'un circuit FPGA Xilinx Virtex II. Le schéma du RNG hybride développé est présenté dans la figure 3-1.

À partir de l'étude réalisée dans le chapitre II, le TRNG sélectionné est celui proposé par Sunar et al. [SMS07]. De plus, conjointement avec l'entreprise de cryptographie et la DCSSI, la solution proposée pour filtrer les défaillances du TRNG est d'utiliser l'implantation matérielle de la batterie FIPS 140-2. Son implantation a été présentée dans la section 2.2-5. Ensuite, la sortie du TRNG, validée en temps réel par le FIPS 140-2, est utilisée pour initialiser un générateur pseudo-aléatoire. Le débit du TRNG est fixé à 6.5 Mb/s. Le PRNG implanté est un AES 128 bits (AES-128). Celui-ci a été développé par l'entreprise. Nous n'avons aucun détail sur les performances de l'implantation.

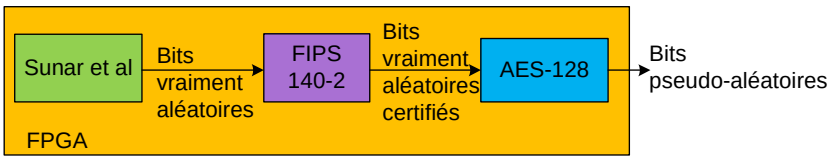

Figure 3-1 – RNG hybride développé conjointement avec l'entreprise spécialisée en cryptographie. Le générateur est composé du TRNG proposé par Sunar et al. [SMS07]. La sortie du TRNG est évaluée en temps réel par la batterie de tests matériels FIPS 140-2. Le germe est ensuite utilisé pour initialiser un PRNG, ici un AES-128.

2 La quête d'un RNG hybride optimal au sein d'un circuit VLSI

Dans certaines applications, une implantation au sein de circuits dédiés (e.g. ASIC) est souvent requise. Si le chapitre II a permis de sélectionner un TRNG au sein d'une plateforme reconfigurable, le générateur à choisir peut être différent au sein d'une solution ASIC. Cette section présente l'implantation de deux RNG hybrides réalisés en technologie CMOS 130 nm. Le premier circuit, OCHREv1 (*On-CHip Randomness Evaluation* version 1), est parti en fabrication en février 2009. Le circuit encapsulé a été reçu en septembre 2009. Cette section présente les premiers résultats de validation du circuit. Le second circuit, OCHREv2 (*On-CHip Randomness Evaluation* version 2), est en cours de développement. Le circuit partira en fabrication en janvier 2010.

2.1 *On-CHip Randomness Evaluation* version 1

2.1-1 Architecture du circuit

L'architecture globale du circuit est présentée à la figure 3-2. Le circuit est composé d'un TRNG, du FIPS 140-2, d'un PRNG et de deux oscillateurs à anneaux.

Le TRNG proposé par Schellekens et al. [SPV06] a été implanté. Celui-ci est une version minimisée de [SMS07], composée de 110 oscillateurs possédant chacun 3 inverseurs. Le TRNG est synchrone avec l'horloge externe $Clk1$. Le générateur est lancé par le signal *reset*. Il est possible de visualiser le signal en sortie de l'arbre de XOR avant échantillonnage grâce au signal *outRingOscillator*.

Le PRNG implanté est l'automate cellulaire mono-dimensionnel à 4 voisins proposé par [Sha02] suivant la règle Ca1d50745. Au sein du circuit visé, le débit maximal du PRNG est de 800 Mb/s. En configurant le signal cmd_{ca}, le générateur peut être initialisé par le TRNG [SPV06] ou par une source de bruit externe.

Les blocs *ringOscillatorsSchellekens33* et *ringOscillatorsSchellekens101* de la figure 3-2 correspondent respectivement à des oscillateurs à anneau composés de 33 et de 101 inverseurs. Les oscillateurs sont indépendants du reste du circuit afin de minimiser les perturbations induites par celui-ci (bruit de couplage, activité). L'implantation des deux RO permet d'étudier le comportement de ceux-ci en fonction des conditions externes exercées sur le circuit (modulation de la tension d'alimentation, variation de la température extérieure). Les deux sorties des oscillateurs sont notées $RO_{33_{\text{out}}}$ et $RO_{101_{\text{out}}}$.

En configurant le signal sel_2, le FIPS 140-2 peut être aussi appliqué sur un générateur externe.

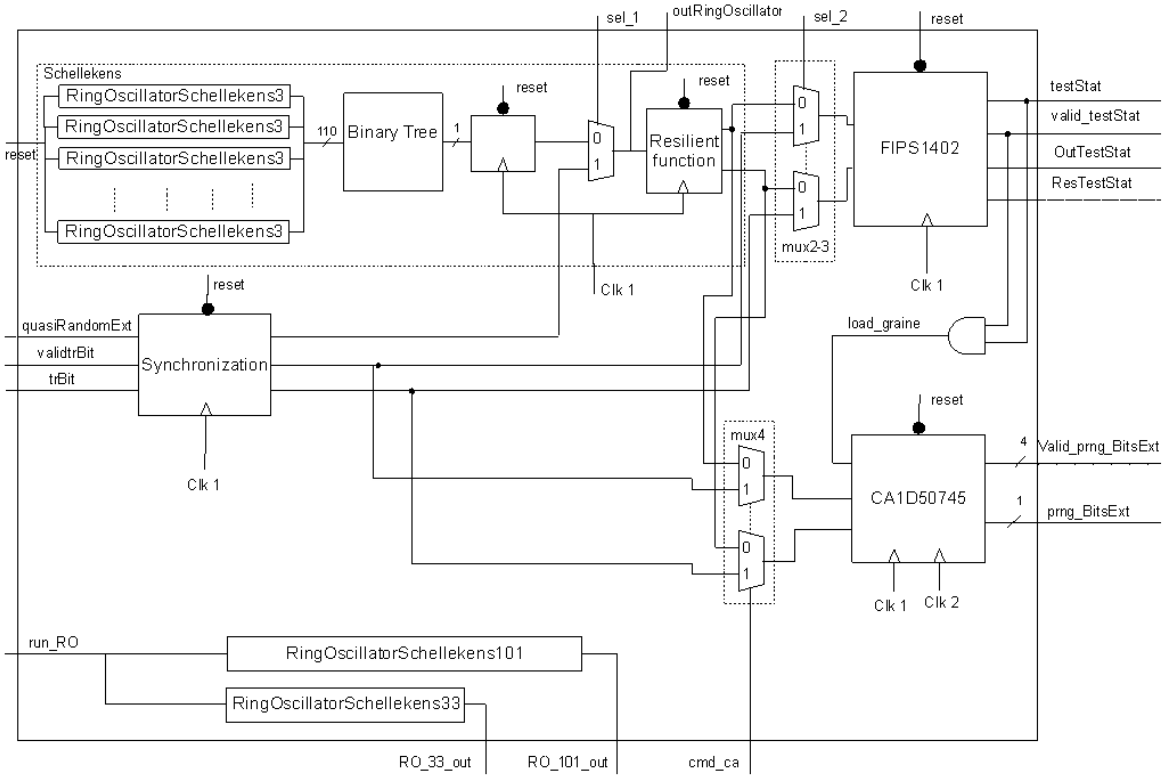

Figure 3-2 – Présentation du circuit OCHREv1.

2.1-2 Surface et consommation du circuit

La surface totale, la puissance dynamique estimée et le chemin critique du circuit sont détaillés dans le tableau 3-1 pour une technologie CMOS 130 nm, 1.2V. La surface du circuit est d'environ 0.35 mm^2. La consommation est très faible (9.04 mW). La fréquence maximale du circuit est de 195 MHz.

Surface totale (μm^2)	Puissance consommée (mW)	Fréquence maximale (MHz)	
		Circuit sans Ca1d50745	Ca1d50745
0.347	9.04	195	819

Tableau 3-1 – Surface totale, puissance dynamique consommée et chemin critique pour une technologie CMOS 130 nm, 1.2V.

2.1-3 Routage du circuit sous l'outil Virtuoso

Le circuit synthétisé et placé-routé est présenté dans la figure 3-3. La surface minimale du circuit pouvant être fabriqué est de 1 mm^2, le circuit possède de grandes parties vides et se retrouve limité par les plots d'entrée/sortie (*pad limited*).

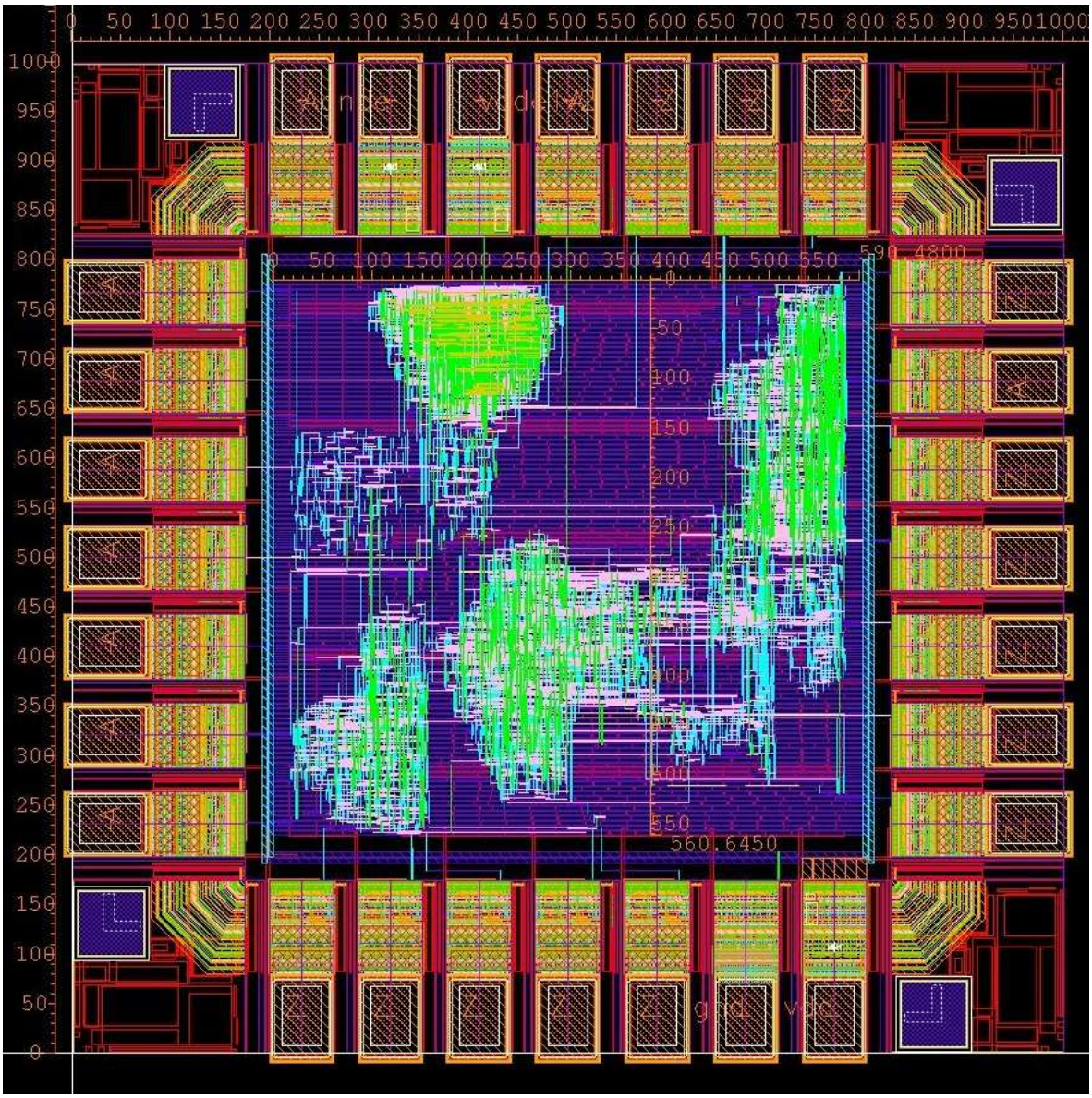

Figure 3-3 – Schéma layout du circuit OCHREv1.

2.1-4 Validation du circuit ASIC

Le chapitre II a présenté la performance de TRNG reconnus au sein d'un circuit FPGA. À notre connaissance, en technologie ASIC, aucune étude de la qualité de TRNG basés

sur l'échantillonnage d'oscillateurs n'a été réalisée. L'objectif de cette sous-section est de présenter les premiers résultats de validation du circuit ASIC. Cette étude est effectuée en trois étapes. Tout d'abord, la source d'aléa du TRNG est évaluée. Ensuite, le bruit échantillonné est présenté. Enfin, les propriétés statistiques de la sortie du TRNG sont examinées.

2.1-5 Analyse de la source de bruit du générateur

Afin d'évaluer la source de bruit du générateur, deux oscillateurs à anneau de longueurs 33 et 101 ont été implantés au sein du circuit ASIC (blocs *ringOscillatorsSchellekens33* et *ringOscillatorsSchellekens101* à la figure 3-2). L'objectif de cette partie est d'évaluer la gigue cycle à cycle produite par les oscillateurs.

Analyse de l'oscillateur à anneau possédant 33 inverseurs

La figure 3-5 présente les caractéristiques de la gigue cycle à cycle de l'oscillateur à anneau possédant 33 inverseurs. La mesure est effectuée à partir de l'observation de 10^6 points échantillonnés à 20×10^6 éch/s. L'horloge analysée possède une fréquence de 216.96 MHz.

L'histogramme montre la distribution de la gigue. En abscisse, l'échelle est égale à 50 ps par division. La valeur moyenne (μ) de la gigue cycle à cycle est de -122.3 fs. La gigue est par conséquent centrée car très proche de 0. Les valeurs maximale et minimale sont respectivement de 98.3 ps et -86.6 ps. L'écart type est de 30.87 ps.

Le tableau 3-2 présente la quantité de gigue observée entre $\mu \pm 1\sigma$, $\mu \pm 2\sigma$ et $\mu \pm 3\sigma$. Par ailleurs, les valeurs théoriques d'une loi normale sont reportées dans le tableau. Ceci permet d'évaluer rapidement si la distribution observée se rapproche d'une loi normale et, pour ce circuit, la gigue cycle à cycle ne suit pas une loi normale.

Oscillateur	Loi normale	RO à 33 inverseurs	R0 à 101 inverseurs
$\mu \pm 1\sigma$	68.26%	66.1%	63.5%
$\mu \pm 2\sigma$	95.44%	99.3%	98.9%
$\mu \pm 3\sigma$	99.74%	100 %	100%

Tableau 3-2 – Répartition de la gigue cycle à cycle entre $\mu \pm 1\sigma$, $\mu \pm 2\sigma$ et $\mu \pm 3\sigma$.

Le spectre de puissance est aussi présenté dans la figure 3-5. En ordonnée, l'échelle est de 12 ps par division. En abscisse, celle-ci est de 10.8 MHz par division. Comme le montre la figure, des composantes fréquentielles sont rencontrées plus fréquemment que d'autres. Par exemple, pour la fréquence de 79.92 MHz, un pic d'amplitude 12 ps est observé. Ceci montre l'existence de parties déterministes au sein de la gigue cycle à cycle.

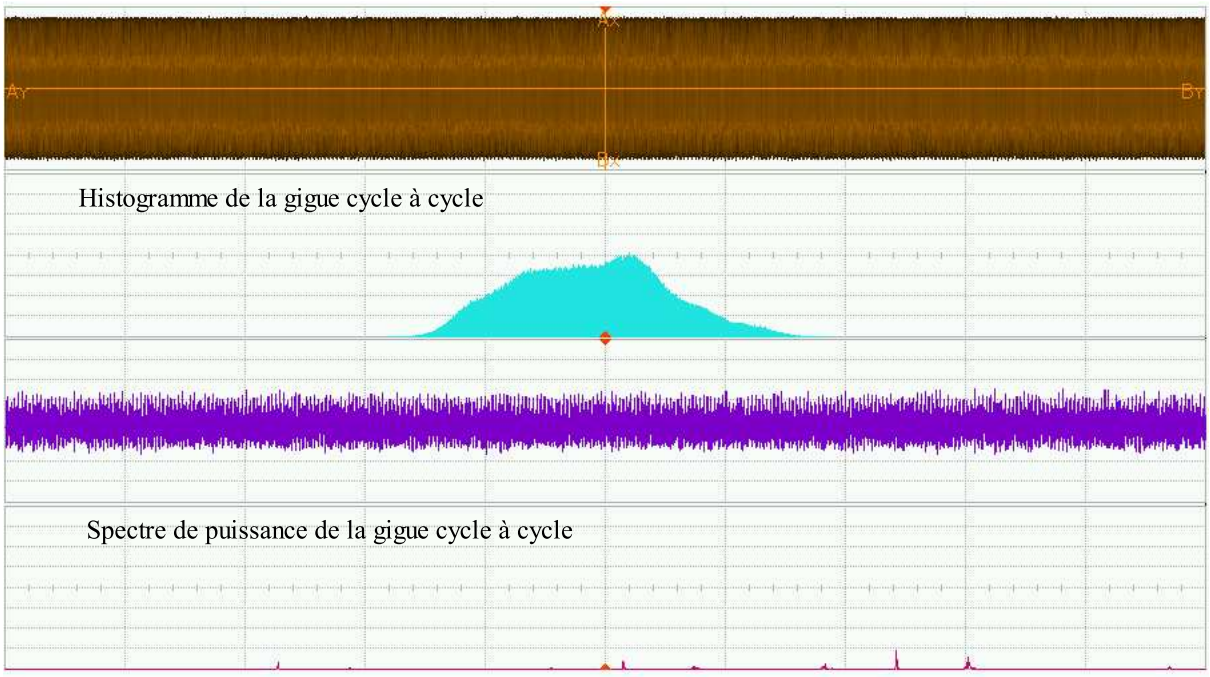

Figure 3-4 – Présentation des caractéristiques de la gigue cycle à cycle de l'oscillateur à anneau possédant 33 inverseurs. La mesure est effectuée à partir de l'observation de 10^6 points échantillonnés à 20×10^6 éch/s. L'horloge analysée possède une fréquence de 216.96 MHz. L'histogramme montre la distribution de la gigue. En abscisse, l'échelle est égale à 50 ps par division. La valeur moyenne (μ) de la gigue cycle à cycle est de -122.3 fs. Les valeurs maximale et minimale sont respectivement de 98.3 ps et -86.6 ps. L'écart type est de 30.87 ps. Pour le spectre de puissance, l'échelle en ordonnée est de 12 ps par division et de 10.8 MHz par division en abscisse.

Analyse de l'oscillateur à anneau possédant 101 inverseurs

La même analyse est réalisée pour l'oscillateur à anneau comportant 101 inverseurs. Les résultats de mesure sont présentés dans la figure 3-5. De la même manière, la mesure est effectuée à partir de l'observation de 10^6 points échantillonnés à 20×10^6 éch/s. La fréquence de l'horloge analysée est de 79.99 MHz.

L'échelle en abscisse de l'histogramme est de 5 ps par division. La valeur moyenne et l'écart type sont respectivement de 85.8 ps et 4.62 ps. Par rapport au RO à 33 inverseurs dans lequel la gigue moyenne est proche de 0 (-112.3 fs) et l'écart type élevé (30.87 ps), ici la gigue observée est relativement faible et non centrée autour de 0. Les valeurs maximale et minimale de la gigue sont respectivement de 104.1 ps et 71.38 ps. Le tableau 3-2 présente la quantité de gigue observée entre $\mu \pm 1\sigma$, $\mu \pm 2\sigma$ et $\mu \pm 3\sigma$. Les statistiques observées montrent que la loi ne s'apparente pas à une loi normale.

Le spectre de puissance est aussi présenté dans la figure 3-5. En ordonnée, l'échelle est de 1.25 ps par division. En abscisse, celle-ci est de 4 MHz par division. La figure montre que pour les fréquences 7, 24.2 et 31.2 MHz, des composantes déterministes sont observées.

La valeur maximale est obtenue pour 31.2 MHz avec un pic de 3.625 ps.

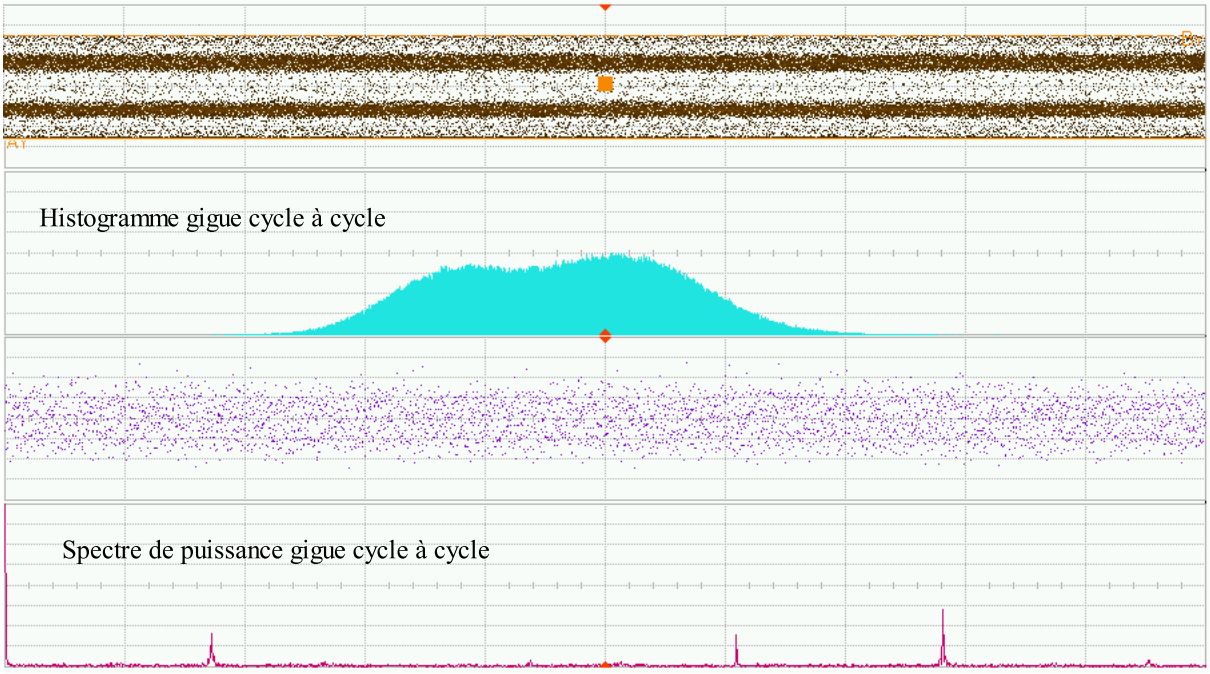

Figure 3-5 – Présentation des caractéristiques de la gigue cycle à cycle de l'oscillateur à anneau possédant 33 inverseurs. La mesure est effectuée à partir de l'observation de 10^6 points échantillonnés à 20×10^6 éch/s. La fréquence de l'horloge analysée est de 79.99 MHz. L'échelle en abscisse de l'histogramme est de 5 ps par division. La valeur moyenne et l'écart type sont respectivement de 85.8 ps et 4.62 ps. Par rapport au RO à 33 inverseurs dans lequel la gigue moyenne est proche de 0s (-112.3 fs) et l'écart type élevé (30.87 ps), ici la gigue observée est relativement faible et non centrée autour de 0. Les valeurs maximale et minimale de la gigue sont respectivement de 104.1 ps et 71.38 ps. Pour le spectre de puissance, l'échelle en ordonnée est de 1.25 ps par division et de 4 MHz par division en abscisse.

2.1-6 Vérification du bon fonctionnement du TRNG

Le fonctionnement du TRNG est vérifié à partir de l'observation du bruit échantillonné, le signal *outRingOscillator* dans la figure 3-2. La figure 3-6 présente un relevé du signal en sortie du circuit. Le signal est synchrone avec l'horloge du TRNG ($Clk1$). Ceci montre le bon fonctionnement du circuit TRNG.

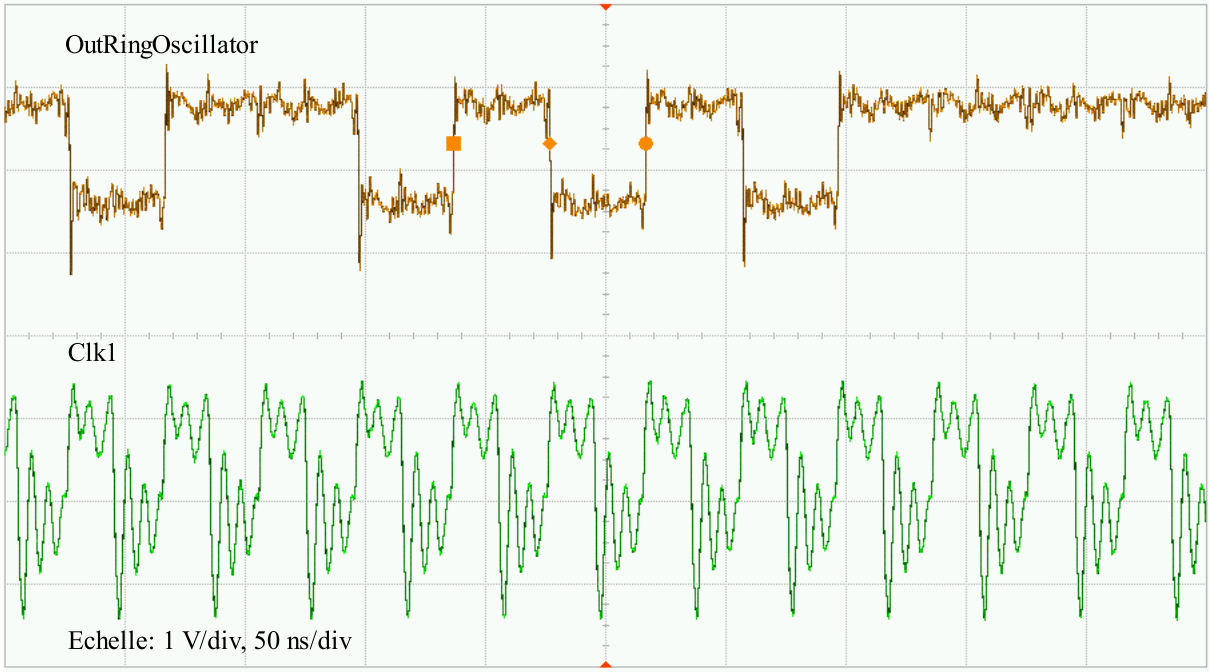

Figure 3-6 – Visualisation du bruit échantillonné (signal *outRingOscillator*) en sortie du TRNG Schellekens et al.

2.1-7 Analyse de la qualité du TRNG

Afin d'évaluer la qualité du générateur, le pourcentage de réussite aux tests FIPS 140-2 est calculé pour des débits de 1.25 Mb/s et 1.875 Mb/s. Le tableau 3-3 résume les résultats obtenus par le générateur. Ceux-ci proviennent de l'analyse de 10^6 séquences binaires composées de 2×10^4 bits. Pour un débit de 1.25 Mb/s, les tests FIPS 140-2 sont réussis par le générateur. Par contre, à partir d'un débit de 1.875 Mb/s, les performances du générateur se dégradent. Le TRNG échoue parfois au *poker*, *run* et *long-run tests*.

Tests	Débit (Mb/s)	
	1.25	1.875
Freq.	100	100
Poker	100	99.85
Run	100	99.7
Long Run	100	99.75

Tableau 3-3 – Résumé de la réussite (en %) du TRNG Schellekens et al. obtenue aux tests FIPS 140-2. Les résultats proviennent de l'analyse de 1M séquences binaires composées de 2×10^4 bits.

2.1-8 Conclusions et perspectives

Après avoir présenté le schéma de principe du circuit, les premiers résultats montrant son fonctionnement correct ont été montrés. À partir de l'observation du bruit échantillonné et de l'analyse de la réussite du générateur aux tests FIPS 140-2, le générateur est validé. Pour un débit de 1.25 Mb/s, le TRNG proposé par Schellekens et al. [SPV06] satisfait l'ensemble des tests du FIPS 140-2. Pour des débits supérieurs, la qualité de la séquence produite par le TRNG se dégrade.

La réception tardive du circuit ASIC au laboratoire n'a pas permis d'étudier en détail les performances du générateur. De plus amples analyses doivent être menées. Il sera nécessaire d'évaluer les propriétés statistiques du bruit échantillonné et deux analyses seront effectuées. La première consistera à utiliser les tests statistiques conseillés par l'AIS 31 afin d'évaluer le signal DAS. La seconde analyse étudiera l'autocorrélation du bruit échantillonné à partir de la mémorisation d'un grand nombre de réalisations du signal. De plus, le circuit sera utilisé pour analyser le comportement du générateur proposé par Schellekens et al. [SPV06] en fonction d'attaques réalisées sur celui-ci (e.g. modulation de la tension d'alimentation).

2.2 *On-CHip Randomness Evaluation* version 2

Le circuit OCHREv1 a permis de prendre en main les outils de conception ASIC. Le fonctionnement correct du circuit permet la réalisation d'un second circuit plus ambitieux : OCHREv2. Plusieurs attentes reposent sur le circuit. Le premier objectif est de comparer les performances de TRNG reconnus au sein d'un circuit ASIC. Le second objectif sera d'étudier le comportement des générateurs face à la réalisation d'attaques. Enfin, le troisième objectif est de produire en sortie des variables aléatoires suivant une loi uniforme, normale et de Rayleigh. Le circuit sera envoyé pour fabrication en janvier 2010. Cette sous-section présente l'architecture et les performances attendues du circuit.

2.2-1 Architecture du circuit

L'architecture globale du circuit est présentée à la figure 3-7. Le circuit est composé des TRNG proposés par Sunar et al. [SMS07], Vasyltsov et al. [VHKK08], Dichtl et al. [DG07] et Yoo et al. [YSKB]. Les tests statistiques proposés par l'AIS 31 et le FIPS 140-2 sont implantés. Le FIPS 140-2 permet de valider en temps réel la qualité de la séquence vraiment aléatoire produite par le TRNG sélectionné. La séquence analysée est ensuite utilisée pour initialiser un générateur pseudo-aléatoire, un AES-128 fonctionnant en mode OFB). Ce mode a été choisi pour ses bonnes performances en tant que PRNG [LS07]. L'AES-128 implanté est distribué librement par l'entreprise ASICS [1]. Tous les 2^{15} cycles, la clé et le message de l'AES sont initialisés par la sortie du TRNG. En sortie de l'AES, deux blocs

1. http ://www.asics.ws/

sont implantés afin de générer des variables aléatoires suivant une loi de Rayleigh et une loi gaussienne (via l'unité Box-Muller). Le fonctionnement des blocs est décrit dans la section 3.

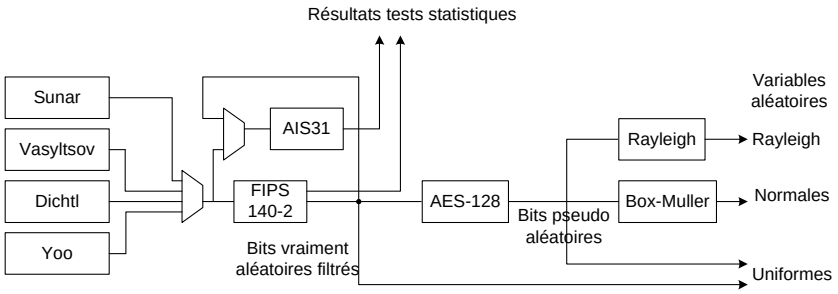

Figure 3-7 – Le circuit est composé des TRNG proposés par Sunar et al. [SMS07], Vasyltsov et al. [VHKK08], Dichtl et al. [DG07] et Yoo et al. [YSKB]. Les tests statistiques AIS 31 et FIPS 140-2 sont implantés. La séquence issue du TRNG est validée par le FIPS 140-2 et sert ensuite à initialiser un générateur pseudo-aléatoire, un AES-128 fonctionnant en mode OFB. En sortie de l'AES, deux blocs sont implantés afin de générer des variables aléatoires suivant une loi de Rayleigh et une loi gaussienne (via l'unité Box-Muller).

2.2-2 Performances

Le tableau 3-4 résume la fréquence maximale des blocs du circuit synthétisés et placés-routés en technologie CMOS 130 nm. Comme le montre le tableau, la fréquence minimale du circuit est obtenue par l'AIS 31. Ce bloc obtient une fréquence maximale de 81 MHz. Les TRNG possèdent des fréquences maximales élevées. Cependant, leur fonctionnement limite le débit de sortie des générateurs à quelques Mb/s. Par conséquent, une analyse en temps réel de la qualité des TRNG par les tests AIS 31 et FIPS 140-2 (fréquence maximale de 177 Mhz) est possible.

La fréquence maximale de l'AES-128 est de 241 MHz. L'AES-128 produit 128 bits tous les 12 cycles d'horloges. Par conséquent, le débit maximal en sortie du bloc est de 2.57 Gb/s.

La fréquence maximale des blocs de Rayleigh et de Box Muller est de 173 MHz. Le principe de fonctionnement et l'implantation des blocs sont présentés dans la section 3. Les débits en sortie des blocs de Rayleigh et de Box Muller sont respectivement de 3.287 Gb/s et 6.5375 Gb/s (car deux nombres gaussiens sont générés à chaque cycle).

Le tableau 3-5 présente la surface occupée et la puissance dynamique consommée par chaque composant du circuit OCHREv2. L'AIS 31 est le bloc occupant la plus grande surface dans le circuit, 0.739 μm^2. La surface du FIPS 140-2 s'élève à 0.068 mm^2, soit environ 10.86 fois moins que l'implantation de l'AIS 31.

Nom du bloc	Fréquence Maximale (MHz)
AIS31	81
FIPS 140-2	177
Dicht	746
Sunar	235
Vasyltsov	628
Yoo07	363
AES	241
Box-Muller	173
Rayleigh	173

Tableau 3-4 – Fréquences maximales des blocs du circuit OCHREv2 synthétisés et placés-routés en technologie CMOS 130 nm.

Les TRNG nécessitent une surface relativement faible. Cependant, le TRNG proposé par Sunar et al. [SMS07] est très coûteux par rapport aux autres générateurs. Sa surface est 11.3, 70 et 2.2 fois plus importante que les TRNG proposés par Dichtl [DG07], Vasyltsov [VHKK08] et Yoo [YSKB].

Nom du bloc	Surface (mm^2)	Puissance consommée (mW)
AIS31	0.739	63
FIPS 140-2	0.068	9.5
Dicht	0.0038	731.7 μW
Sunar	0.043	11.4
Vasyltsov	0.000613	529.2μW
Yoo	0.019	5.2
AES	0.111	14.6
Box-Muller	0.17	4
Rayleigh	0.008	1.8

Tableau 3-5 – Surface et consommation dynamique des blocs du circuit OCHREv2 synthétisés et placés-routés en technologie CMOS 130 nm.

2.2-3 Discussions et perspectives

Le circuit OCHREv1 a permis de prendre en main les outils de conception ASIC. Le circuit OCHREv2 est une suite logique au circuit OCHREv1. Celui-ci permettra d'étudier la qualité de quatre TRNG implantés sur cible ASIC. Le comportement des générateurs sera analysé en fonction de l'environnement extérieur dans lequel le circuit sera utilisé.

De plus, le circuit permettra de générer des variables aléatoires suivant des lois uniformes, normales et de Rayleigh.

Après avoir présenté le schéma global du circuit, les performances obtenues en technologie CMOS 130 nm ont été développés. Les blocs Box-Muller et Rayleigh n'ont pas encore été expliqués. L'objectif de la section suivante est d'expliquer leur implantation.

Le layout du circuit est présenté dans la figure 3-8. Une mémoire RAM de taille 1024×16 bits à été générée par STMicroelectronics. Celle-ci est utilisée pour le test T5 de l'AIS 31. La surface du circuit sera environ de $4mm^2$.

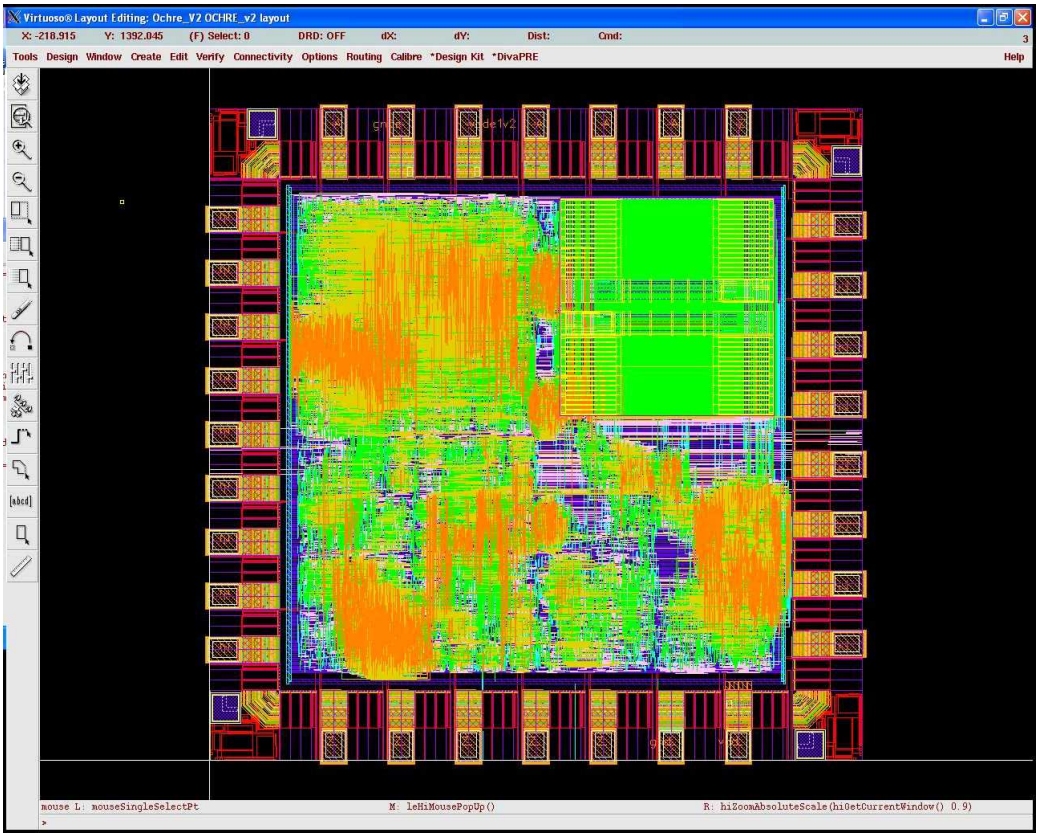

Figure 3-8 – Schéma layout du circuit OCHREv2.

3 Vers un RNG gaussien et de Rayleigh à très haut débit

La section précédente a présenté l'implantation matérielle d'un générateur uniforme au sein de circuits FPGA et ASIC. Les générateurs hybrides permettant de générer des nombres aléatoires uniformes sont très utilisés dans les applications cryptographiques. Cependant, dans certaines applications, il est parfois nécessaire de générer des nombres

aléatoires suivant une distribution de probabilité différente. Par exemple, comme le présente la figure 3-9, en communication numérique, il est souvent nécessaire de connaître le taux d'erreur binaire d'une chaîne de transmission. Le taux d'erreur binaire correspond à une mesure relative du nombre d'erreur binaire mesuré en sortie d'une transmission numérique. Les erreurs sont généralement engendrées par le canal de transmission [Pro00].

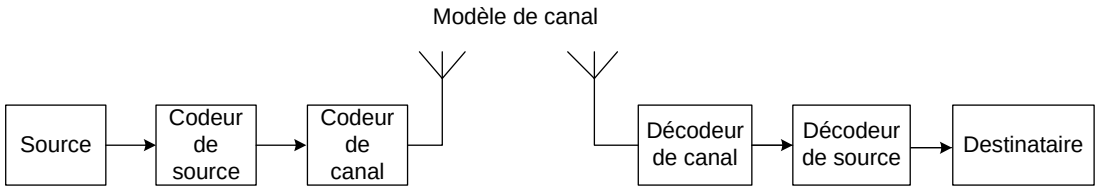

Figure 3-9 – Système de communication numérique avec processus de compression des données et de protection contre les erreurs de transmission.

Des modèles de canaux de transmission sont souvent utilisés pour simuler le comportement de tels systèmes. Cependant, les taux d'erreur binaire actuels (10^{-9}) entraînent la réalisation d'un grand nombre de simulations. Par exemple, pour une confiance de $1-C$, une précision relative égale à r, l'observation d'un événement de probabilité p nécessite au moins

$$N_b \geq 2\left(\text{erfc}^{-1}\left(C\right)\right)^2 \frac{1}{r^2}\left(\frac{1}{p}-1\right) \tag{3-1}$$

réalisations [BDI86]. Par exemple, pour $r = 10^{-2}$, $C = 10^{-4}$ et $p = 10^{-9}$, au moins 1.51×10^{14} simulations sont nécessaires.

La complexité des modèles de canaux de transmission engendre des temps de calcul très importants. Une alternative pour améliorer la rapidité des simulations (souvent des simulation de Monte-Carlo [BDG03]) est d'implanter le modèle de canal de transmission en matériel, au sein de circuits reconfigurables.

En communications numériques, un modèle de canal de transmission souvent utilisé est un canal avec bruit gaussien additif. Celui-ci nécessite l'utilisation d'un générateur de bruit gaussien. Des solutions existent dans la littérature. Cependant, les simulations numériques reposent sur la qualité aléatoire d'un grand nombre d'échantillons. La déviation de la distribution de probabilité de la variable aléatoire générée par rapport à une distribution gaussienne idéale peut avoir de grandes répercussions sur les résultats des simulations numériques [LLV+05]. Par conséquent, la qualité des variables aléatoires produites est un point primordial.

Une des limitations des générateurs aléatoires gaussiens (GRNG) est la génération des échantillons situés dans la queue de la distribution de probabilité de la v.a. gaussienne. Même si ces échantillons sont rarement produits, ceux-ci sont très importants dans les

simulations numériques. Ne pas les produire peut engendrer de graves répercussions sur les résultats de simulation [Lee06].

Cette section présente la conception matérielle du circuit permettant la simulation, temps réel d'un canal à bruit gaussien additif et d'un canal de Rayleigh. Dans un premier temps, un état de l'art des techniques de génération des variables aléatoires gaussiennes est effectué. Ensuite, l'implantation d'un algorithme existant est présentée.

3.1 Techniques de génération

Si il était possible d'utiliser des sources de bruits analogiques idéales afin de produire des échantillons suivant une loi de probabilité normale, le débit du générateur serait trop faible. Par conséquent, des solutions numériques sont introduites. Les GRNG sont classés en quatre familles présentées dans cette section.

3.1-1 Variables aléatoires gaussiennes

Les variables aléatoires gaussiennes sont souvent générées à partir de variables aléatoires (v.a.) uniformes prenant leurs valeurs dans l'intervalle $]0,1[$. Celles-ci sont notées ici $\mathcal{U}(0,1)$ ou directement $\mathcal{U}$. Certains algorithmes utilisent plusieurs variables aléatoires uniformes. Par exemple, si deux v.a. indépendantes uniformes sont requises, celles-ci sont notées U_0 et U_1.

Une variable aléatoire suit une loi normale de paramètre μ (réel) et σ (réel supérieur à zéro), si X admet la densité de probabilité

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}. \tag{3-2}$$

Les variables aléatoires générées par les différentes méthodes suivent généralement une loi normale centrée réduite. Par conséquent, par la suite, X est considérée comme étant de moyenne nulle et d'écart type égal à un. La densité de probabilité de la v.a. X est donnée par

$$f_X(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}. \tag{3-3}$$

La fonction de répartition de X peut être exprimée à partir de la fonction d'erreur

$$F_X(x) = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right) = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right), \tag{3-4}$$

où erf désigne la fonction d'erreur et est donnée par

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt. \tag{3-5}$$

La partie suivante décrit les méthodes de génération des variables aléatoires gaussiennes.

3.1-2 Méthode par transformation

Plusieurs méthodes de transformation existent dans la littérature. Les plus répandues sont les transformations B-Muller [BM58], d'inversion et de rejet-acceptation.

Méthode Box-Muller

La méthode Box-Muller [BM58] (BM) génère à partir de deux variables aléatoires uniformes et indépendantes U_0, U_1, deux variables aléatoires X_0, X_1. Les opérations réalisées sur U_0 et U_1 sont les suivantes

$$f(u_0) = \sqrt{-2\ln(u_0)}, \tag{3-6}$$

$$g_0(u_1) = sin(2\pi u_1), \tag{3-7}$$

$$g_1(u_1) = cos(2\pi u_1), \tag{3-8}$$

$$x_0 = f(u_0) \times g_0(u_1), \tag{3-9}$$

$$x_1 = f(u_0) \times g_1(u_1). \tag{3-10}$$

Plusieurs implantations matérielles de la méthode Box-Muller sont présentées dans la littérature. La première contribution a été introduite par Boutillon et al. dans [BDG03]. Le générateur utilise une combinaison de la méthode Box-Muller et du Théorème Centrale Limite (TCL)). Le TCL permet de réduire les erreurs d'approximation lors du calcul des fonctions rencontrées dans l'algorithme BM. L'inconvénient est la réduction du débit du GRNG par un facteur 4. Le circuit, implanté au sein d'un FPGA Altera Flex 10K1000EQC240-1 occupe une surface de 437 cellules logiques et atteint un débit de 24.5 millions d'échantillons par seconde. La fréquence maximale du circuit est de 98 MHz.

A partir des travaux réalisés dans [BDG03], Xilinx a réalisé un bloc IP du GRNG [Xil02]. De même, Fung et al. dans [EKN+04] présente une implantation ASIC en technologie $0.18\mu m$ du circuit [BDG03] et obtient un débit égal à 182 millions d'échantillons par seconde. De plus, [EKN+04] a réalisé une implantation du GRNG au sein d'un FPGA Virtex II XC2V1000-6. Le débit est de 245 millions d'échantillons par seconde. Malgré les performances de [BDG03], la valeur maximale des échantillons est limitée à 4σ. De plus, [LLVC03a] montre la faible qualité statistique des échantillons générés.

Plus récemment, [Lee06] propose une implantation de la méthode Box-Muller. A la différence de leurs travaux précédemment présentés dans [LLVC04], la technique proposée n'utilise pas de TCL. Ainsi, le débit du GRNG est augmenté. Dans [LLVC04], l'importance des erreurs de quantification et d'approximation oblige à utiliser le TCL pour améliorer

la qualité statistique de la séquence produite. La valeur maximale des échantillons est de 6.7σ. Les échantillons produits par [Lee06] atteignent les 8.2σ. Au sein d'un FPGA Xilinx Virtex 4 XC4VLX100-12, le GRNG utilise 1452 slices (2.25 cellules logiques par slice), 3 blocs RAM de 18Kb et 12 blocs DSP48. Pour une fréquence de fonctionnement du circuit égale à 375 MHz, le débit du générateur est égal à 750 millions d'échantillons par seconde. [Lee06] génère les variables aléatoires uniformes à partir du générateur de nombres aléatoires uniformes de Tausworthe [Tau65]. D'après [Lee06], le générateur est rapide et occupe une surface moins importante qu'un générateur de type LFSR (comme utilisé dans [BDG03]) tout en étant de meilleure qualité. Le générateur de Tausworthe utilisé correspond à l'algorithme présenté dans [L'E96]. Le générateur est constitué d'une combinaison de trois PRNG de type LFSR. Si les performances de [Lee06] semblent alléchantes en termes de débit et de valeur maximale, le GRNG possède une complexité importante en surface, ce qui limite son intérêt.

Réaliser en matériel la méthode BM impose l'implantation et l'évaluation des fonctions logarithmique, racine carrée, sinus et cosinus. A partir de l'étude de la fonction de probabilité de la loi normale $\mathcal{N}(0,1)$ et du nombre d'échantillons désiré (10^{15}), [Lee06] calcule la valeur maximale de x_0 et x_1 à générer. Ainsi, les échantillons doivent être générés jusqu'à 8.1σ pour une population de 10^{15} échantillons. Ensuite, le nombre de bits requis par U_0 est calculé. La taille de U_0 permet de déterminer la précision désirée en sortie du GRNG.

Dans [Lee06], une précision de 48 bits est requise. Le nombre de bits pour U_1 est fixé arbitrairement à 16. L'objectif de [Lee06] est de produire des échantillons gaussien de taille 16 bits avec une précision de 11 bits. Cette précision a été choisie à partir du nombre de bits consacré à la partie fractionnaire. La partie entière de la valeur maximale de x_0 et x_1 est représentée par 5 bits. Par conséquent, 11 bits sont dédiés à la partie fractionnaire. Les fonctions sont approximées par des polynômes minimax via Maple et par l'algorithme de Remez. A partir d'une précision minimale, le domaine des fonctions est découpé en intervalles uniformes. Dans chaque segment, les polynômes sont calculés en utilisant le schéma de Horner. Au sein du graphe flot de signal (GFS), [Lee06] utilise une réduction de la précision par arrondi. La précision des coefficients des polynômes est obtenue en analysant de manière exhaustive l'erreur en sortie du GFS. Des polynômes de degré un sont choisis pour l'approximation des fonction sinus/cosinus et racine carrée. Des polynômes de degré deux sont sélectionnés pour la fonction logarithmique. La taille totale de la mémoire pour sauvegarder les coefficients est de 24 704 bits, requérant trois BRAM de taille 18 Kb. En dehors de la surface du GRNG, une critique de [Lee06] est la méthode d'analyse de la précision de l'algorithme Box-Muller. De plus, une étude plus complète de la qualité statistique des échantillons produits aurait été souhaitable.

Plus récemment, une implantation de la méthode Box-Muller est proposée dans [AFCS08]. Le générateur a été précédemment introduit dans [ABS05]. La fonction $f(u_0)$ est approchée à partir de la méthode introduite dans [Mat07]. La fonction est approchée par un polynôme idéal. Le principe de la méthode utilisée est similaire à celle employée par [Lee06]. Au sein

	[BDG03]	[Xil02]	[LLVC04]	[ABS05]	[Lee06]	[AFCS08]
FPGA ciblé	Altera Flex 10KE	Xilinx Virtex-II XC2V4000-6				
Cellules logiques	437	480	2514	702	1528	534
Blocs RAM	0.5	5	2	5	3	2
MULT 18x18	Inconnu	5	8	1	12	3
% occupation	8%	2%	10.9 %	3%	6.6%	2.3%
fréquence max.	98	245	133	165	233	248
Débit (Mbits/s)	24.5	245	133	165	466	496
$\sigma_{\max}$	4σ	4.8σ	6.7σ	5σ	8.2σ	6.66 σ

Tableau 3-6 – Comparaison de la performance de GRNG utilisant la méthode BM.

d'un FPGA Xilinx Virtex-II XC2V4000-6, la fréquence maximale du circuit est de 248 MHz, générant ainsi 496 millions de variables aléatoires par seconde. La valeur maximale d'un échantillon est de 6.66σ. Le circuit occupe 534 slices, deux blocs RAM et trois blocs DSP.

Dans la méthode Box-Muller, la précision du générateur repose sur la qualité d'implantation de la fonction $f(u_0)$ et sur le nombre de bits consacré à la partie fractionnaire de U_0. La technique généralement utilisée est d'approcher la fonction par un polynôme idéal minimisant l'erreur d'approximation. Cependant, un des problèmes délicat est le découpage en segments du domaine de la fonction. Un nombre de segments faible imposera pour la même précision d'utiliser des polynômes d'approximation d'ordres plus élevés. Pour la même latence, ceci entraîne une augmentation du nombre de ressources. Un compromis entre la taille de la mémoire nécessaire à la mémorisation des coefficients et le nombre de ressources arithmétiques doit être trouvé. Plusieurs études ont été menées sur cette problématique [CLLV07], [Max60] et [PM07].

Dans [BDG03] et [LLVC04], la fonction $f(u_0)$ est segmentée de manière non uniforme. Malgré ceci, ces GRNG utilisent le TCL afin d'améliorer la qualité statistique de la séquence produite. [Lee06] améliore la précision du générateur proposé en découpant de manière hiérarchique le domaine de la fonction. A la différence, [AFCS08] propose une segmentation hybride dans laquelle un découpage logarithmique et uniforme combinés est utilisé.

Le tableau 3-6 résume les coûts et performances des GRNG utilisant la méthode BM.

Méthode par inversion

La méthode par inversion permet de générer des variables aléatoires gaussiennes à partir de la fonction de répartition inverse de la loi normale

$$F_X^{-1}(u) = \sqrt{2}\mathrm{erf}^{-1}\left(2u-1\right), \tag{3-11}$$

où $u \in \mathcal{U}(0,1)$. L'algorithme est basé sur le fait qu'un échantillon x, de fonction de répartition F_X peut être obtenu à partir de $x = F_X^{-1}(u)$.

Peu d'implantations de la méthode par inversion existent dans la littérature. La précision de la méthode repose sur l'implantation de la fonction de répartition inverse de la loi normale $F_X^{-1}(u)$. Cette fonction, de par ses asymptotes verticales (pour les valeurs de u proches de 0 ou 1), est difficile à implanter. Par exemple, [CMB04] utilise une table pour mémoriser les résultats de la fonction. Le domaine de la fonction est découpé en segments de taille uniforme. Cependant, peu d'informations sur la taille de la mémoire et du GFS à utiliser sont fournies. De plus, aucune évaluation de la valeur maximale atteinte par les échantillons générés et leurs propriétés statistiques n'est effectuée.

Afin de diminuer la taille de la mémoire requise, [LCVL06] décompose le domaine de la fonction $F_X^{-1}(u)$ en intervalles non-uniformes. La symétrie de la fonction autour de $u = 0.5$ est utilisée. Seule la fonction $x = |F_X^{-1}(u/2)|$ est approchée en utilisant les coefficients de Chebyshev [Mul06]. La valeur maximale générée en sortie ($\sigma_{\max}$) est fonction du nombre de bits de l'entrée U. Soit B_u, la largeur en bit de U. La valeur maximale générée est obtenue par

$$\sigma_{\max} = \left|F_X^{-1}\left(2^{-B_u-1}\right)\right| = F_X^{-1}\left(1 - 2^{-B_u-1}\right). \tag{3-12}$$

À partir de la période du générateur désirée (N), [LCVL06] détermine la taille minimale de B_u et la valeur maximale $\sigma_{\max}$ nécessaire. [LCVL06] choisit $\sigma_{\max}$ de telle sorte que pour N échantillons générés, la probabilité que l'un d'eux soit supérieur à $\sigma_{\max}$ est inférieure à $1/2$, soit

$$N \times \left(2 \times F_X\left(-\sigma_{\max}\right)\right) < \frac{1}{2} \tag{3-13}$$

$$B_u > -\log_2\left(\frac{1}{4N}\right) - 1. \tag{3-14}$$

Ainsi, pour $N = 2^{15}$ et des échantillons gaussiens de taille 16 bits, B_u doit être égale à 51 bits, résultant en un $\sigma_{\max}$ égal à 8.1. En pratique, [LCVL06] prend $B_u = 52$ afin d'avoir un $\sigma_{\max}$ égal à 8.2 et de pouvoir comparer le générateur avec [Lee06]. Il est nécessaire de coder la partie entière de la variable gaussienne générée X sur 5 bits. Ainsi, les 11 bits restants sont dédiés à la partie fractionnaire. Une précision de 11 bits est donc requise en sortie du générateur.

[LCVL06] utilise un découpage non-uniforme de la fonction à approcher. La fonction est segmentée suivant la méthode introduite dans [LLVC03b]. Comme il sera expliqué dans la suite, le schéma de segmentation est déterminé par la valeur du coefficient directeur de la fonction à la fin de l'intervalle analysé. Le domaine de la fonction ainsi découpé est composé de 144 segments. L'étage d'indexage de la mémoire des coefficients nécessite les 51 bits de poids forts de l'entrée U.

La précision du GFS est analysée par la technique *MiniBit* introduite dans [LAGLM05]. La taille des opérateurs et des signaux est limitée par l'utilisation de troncature. En sortie du générateur, un arrondi au plus proche est réalisé afin d'améliorer la précision du GRNG. Le bruit total engendré en sortie est exprimé en fonction des différents troncatures et arrondis. La méthode utilisée est criticable car [LCVL06] suppose que l'erreur engendrée après

chaque troncature est maximale. Le nombre de bits pour représenter les parties fractionnaires est calculé à partir d'un problème d'optimisation. La fonction de coût correspond à l'estimation de la surface des composants principaux d'un FPGA : mémoire, additionneur et multiplieur. La taille totale de la mémoire permettant de mémoriser les coefficients et des mémoires indexant celle-ci est égale à 7432 bits.

Méthode par récurrence ou méthode de Wallace

Le principe de l'algorithme de Wallace, introduit dans [Wal96], est d'appliquer des transformations linéaires sur un ensemble de variables aléatoires gaussiennes. Cet algorithme permet d'atteindre des débits très élevés. De plus, aucune fonction telle que $\sqrt{x}$, $\log(x)$, $\cos(x)$ ou $\sin(x)$ n'est requise. À partir d'un ensemble de K variables aléatoires normales, un vecteur X est créé. Un nouveau vecteur de K variables aléatoires normales est obtenu en calculant

$$X^{(1)} = AX, \tag{3-15}$$

où A est une matrice orthogonale. Si les K variables d'initialisation sont indépendantes et suivent une loi normale centrée réduite $\mathcal{N}(0,1)$, il en sera de même du vecteur $X^{(1)}$. De plus, la transformation préserve la somme des carrés.

L'implantation présentée par [Wal96] utilise deux urnes de dimension KL. La première contient les variables aléatoires gaussiennes d'initialisation. La seconde urne permet de conserver les variables aléatoires générées. A chaque étape de génération, la seconde urne est remplie en réalisant un ensemble de L transformations successives. A chaque transformation, un vecteur de K variables est pris dans la première urne afin d'être multiplié par la matrice A. À la fin des L transformations, le contenu de la première urne est remplacé par celui de la seconde et les variables aléatoires sont mises à disposition de l'utilisateur.

Dans [Wal96], la méthode utilise $K = 4$, $L = 256$, soit un ensemble de $N = 1024$ variables aléatoires. Dans l'urne contenant les anciennes variables aléatoires, celles-ci sont rangées en ligne dans une matrice de dimension $L \times K$. Dans la seconde urne, les variables aléatoires générées sont rangées en colonne dans une matrice de dimension $L \times K$. Ceci permet de transposer les valeurs dans l'urne à la suite de L transformations. Si L est impair, la transposition suffit pour assurer un réarrangement des variables aléatoires. Cependant, si L est pair, la transposition n'est pas suffisante. [Wal96] corrige ceci en débutant le remplissage de l'urne contenant les anciennes variables aléatoires à une autre ligne que celle d'index zéro. De plus, à chaque remplissage d'une nouvelle ligne de la matrice, le pointeur sur la nouvelle ligne est incrémenté d'une valeur impaire différente de un modulo la valeur de L ($L = 256$).

En pratique [Wal96] utilise un générateur pseudo-aléatoire uniforme. Au début de chaque nouvelle série de L transformations, celui-ci a pour rôle d'initialiser la valeur du pointeur (entre 1 et 255) sur la matrice contenant les anciennes variables aléatoires, de fournir une valeur impaire différente de un permettant l'incrémentation de l'index de ligne sur la matrice entre deux transformations et enfin de choisir aléatoirement une des quatre

matrices A à utiliser. De plus, le générateur uniforme permet d'initialiser l'urne lors du premier appel au GRNG.

[Wal96] conseille d'utiliser la méthode Box-Muller pour la qualité des échantillons générés. Les variables d'initialisation sont normalisées afin que la valeur moyenne des variables élevées au carré soit égale à 1. La matrice A étant orthogonale, les transformations ne modifient pas la valeur de la somme des carrés. Ceci implique que la somme (χ^2_N) des variables aléatoires élevées au carré est toujours constante quel que soit le nombre de transformations réalisées. Afin de corriger ceci, une variable aléatoire parmi les N générées est utilisée et approchée pour que celle-ci suive la loi de probabilité χ^2_N. La variable est utilisée afin de corriger les $N-1$ variables aléatoires de l'urne générée.

L'inconvénient de la méthode de Wallace est le phénomène de rebouclage. Celui-ci peut engendrer des corrélations entre les sorties des transformations successives. Cependant, il est possible de minimiser la corrélation en choisissant correctement les paramètres du générateur comme la taille des urnes utilisées ou celle de la transformation [LLV+05].

[LLV+05] présente une implantation de la méthode de Wallace. Les paramètres du générateur sont choisis afin de minimiser la corrélation. D'après [LLV+05], la qualité du générateur développé est meilleure que les circuits [LLVC04] et [BDG03] utilisant la méthode Box-Muller et le TCL.

Méthode par rejet-acceptation

La méthode par rejet-acceptation suppose qu'il est possible d'obtenir une variable aléatoire X de loi de probabilité $f(x)$ à partir d'une loi de probabilité $g(x)$ facilement simulable. Si

$$\forall x, \exists c | f(x) \leq c.g(x), \tag{3-16}$$

il est possible de générer la variable aléatoire X suivant la loi de probabilité $f(x)$.

Dans un premier temps, il est nécessaire de déterminer une loi de probabilité g et une constante c tel que la condition (3-16) soit acceptée pour tout x sur l'intervalle de f. L'algorithme se divise en deux phases et est décrit ci-dessous.

```
1  1. Simuler
2     Tirer aléatoirement Y de loi de probabilité g
3      Tirer aléatoirement U de loi de probabilité g
4  2. Evaluation de la variable aléatoire Y
5       Si U<(f(Y))/(c.g(Y))
6       Alors
7        X = Y
8       Sinon
9        retourner à 1
10      Fin de Si
```

Les deux algorithmes les plus utilisés en VLSI utilisant la méthode par rejet-acceptation sont la méthode par coordonnées polaires [FZC04] et celle de Ziggurat [MT00], [ZLL+05]. Cependant, ces algorithmes ne permettant pas d'assurer un débit constant, leur utilisation est limitée au sein des circuits VLSI.

Evaluation de la qualité des échantillons aléatoires gaussiens générés

Plusieurs méthodes sont envisageables afin de mesurer l'adéquation de la variable aléatoire normale générée $X(x)$ par le GRNG avec une loi normale centrée réduite idéale $\mathcal{N}(0,1)$. [BDG03] utilise l'erreur relative ($\xi_{X(x)}$) entre la fonction de probabilité (pdf) de X et celle de la loi normale $\mathcal{N}(0,1)$

$$\xi_{X(x)} = \frac{X(x) - \mathcal{N}(0,1)(x)}{\mathcal{N}(0,1)(x)}. \tag{3-17}$$

De plus, il est possible d'utiliser des tests statistiques similaires à ceux utilisés pour tester les propriétés statistiques d'une variable aléatoire uniforme. Par exemple, le test d'adéquation du χ^2, le test Kolmogorov-Smirnov ou encore le test Anderson-Darling sont appropriés pour cette analyse. L'avantage du test d'Anderson-Darling est d'évaluer plus en détails la queue de la fonction de répartition de la variable aléatoire générée.

3.1-3 Sélection d'un algorithme permettant de générer des nombres aléatoires gaussiens à très haut débit

Un état de l'art de la qualité des GRNG est présenté dans [TL07]. Le choix de la méthode permettant de générer des variables aléatoires gaussiennes est fonction de la simplicité de l'algorithme, de l'erreur engendrée par son implantation (approximation), de la qualité statistique des échantillons générés et enfin de son efficacité (débit, latence, surface). D'après [AFCS08], le débit d'un algorithme utilisant la méthode de l'inversion est moitié moindre que celui utilisant la méthode BM.

Le tableau 3-7 résume les performances des GRNG les plus performants des quatre méthodes de génération. Celui-ci indique que la technique de l'inversion est à privilégier pour maximiser la valeur de $\sigma_{\max}$. Cependant, à surface égale, la technique Box-Muller est plus performante en terme de débit atteint. Le débit du générateur [AFCS08] est environ 2.13 fois plus rapide que [LCVL06]. Sous l'hypothèse que les variables normales générées par [AFCS08] soient bien indépendantes, il serait possible de sommer des paires de v.a. afin d'obtenir une valeur maximale $\max_\sigma = 13.32$. Par conséquent, cette technique est privilégiée dans la suite de ce travail.

Par ailleurs, la méthode présente un autre avantage. Un objectif de la thèse est de produire des variables aléatoires de Rayleigh. Une variable aléatoire de Rayleigh ($\mathcal{R}$) de paramètre σ est obtenue à partir d'une variable aléatoire uniforme $\mathcal{U}_0 \in [0;1]$ en appliquant la transformation

$$r(u_0) = \sigma\sqrt{-2\ln(u_0)}. \tag{3-18}$$

Ceci revient à utiliser la fonction $f(u_0)$. Par conséquent, une partie de l'algorithme Box-Muller permet de générer aussi des variables aléatoires de Rayleigh. Il suffira ensuite à l'utilisateur de multiplier les résultats obtenus par une constante propre à l'application visée.

	[FZC04]	[ZLL+05]	[LLV+05]	[LCVL06]	[AFCS08]
Méthode	Polaire	Ziggurat	Wallace	Inversion	Box-Muller
FPGA ciblé	Altera Mercury EP1M120	Xilinx Virtex-II XC2V4000-6			
Cellules logiques	336	891	770 (slices)	548 (slices)	534 (slices)
Blocs RAM	1.4	4	6	2	2
MULT 18x18	Inconnu	2	4	2	3
Fréquence max.	73	170	155	232	248
Débit (Mbits/s)	146	168	155	232	496
$\sigma_{\max}$	Inconnu	Inconnu	7σ	8.2σ	6.6σ

Tableau 3-7 – Comparaison des techniques de GRNG.

La partie suivante présente en détail l'implantation de l'algorithme Box-Muller proposée récemment dans [AFCS08].

3.1-4 Présentation du GRNG proposé par Alimohammad et al.

Alimohammad et al. montrent dans [AFCS08] que la valeur maximale ($\max_\sigma$) en sortie d'un GRNG utilisant la méthode BM est fonction du nombre de bits de u_0 (b_{u_0}). Ceci s'explique car les valeurs maximales de $g_0(u_0)$ et $g_1(u_0)$ sont égales à 1. Par conséquent, les valeurs maximales de x_0 et x_1 sont déterminées par le calcul de $f(u_0) = \sqrt{-2\ln(u_0)}$. Plus la valeur de u_0 est faible, plus la valeur de $\max_\sigma$ est élevée. Celle-ci est déterminée par

$$\sigma_{\max} = \sqrt{2b_{u_0}\ln(2)}. \tag{3-19}$$

Par exemple pour $b_{u_0} = 32$, la valeur maximale $\max_\sigma$ atteinte par le GRNG est égale à 6.66σ. Par conséquent, la précision du GRNG dépend de l'implantation matérielle de la fonction f. [AFCS08] propose d'approcher la fonction $f(u_0)$ par un polynôme d'approximation.

Cependant, il est nécessaire de segmenter la fonction en plusieurs intervalles. La fonction $f(u_0)$ est présentée dans la figure 3-10. Le coefficient directeur de $f(u_0)$ est élevé lorsque u_0 est proche de 0 ou de 1.

Lorsque u_0 est proche de 0 (respectivement 1), la précision du calcul a un impact sur les échantillons situés en queue de gaussienne (respectivement au milieu de la distribution de probabilité). Par conséquent, une implantation efficace et précise de la fonction $f(u_0)$ doit être réalisée. [AFCS08] effectue une segmentation hybride de la fonction. Celle-ci utilise une décomposition logarithmique et uniforme. La segmentation est présentée dans la figure 3-11.

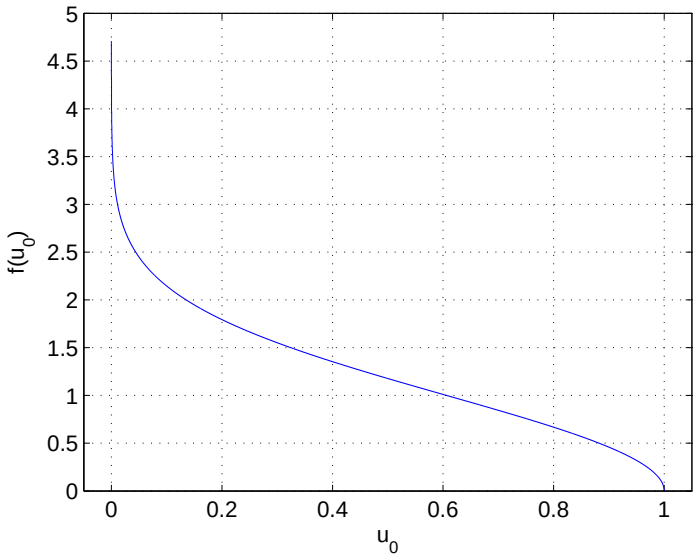

Figure 3-10 – Fonction $f(u_0)$.

Dans un premier temps, le domaine est divisé en deux intervalles, $r_0 \in]0, 0.5[$ et $r_1 \in [0.5, 1[$. La valeur de u_0 est codée en virgule fixe sur 32 bits. Les intervalles r_0 et r_1 sont ensuite divisés en b_{u_0} sous segments. Le découpage est réalisé de manière logarithmique. Le nombre total de segments est égal à $2 \times (b_{u_0} - 1)$. Chaque segment est dénoté $s_{r,w}$, où $r \in \{0, 1\}$ et $w \in \{0, \ldots, b_{u_0} - 2\}$ désignent respectivement le premier intervalle d'appartenance (r_0, r_1) et le numéro du segment logarithmique. Ensuite, les segments $s_{r,w}$ sont divisés en $L = 2^l$ sous-intervalles uniformes. Ceux-ci sont notés $s_{r,w,l}$.

Des polynômes de degré 1 de la forme

$$p(u_0) = a \times x + b \tag{3-20}$$

sont utilisés et les polynômes minimisant l'erreur d'approximation au sens des moindres carrés sont alors recherchés grâce à la boîte à outils de Matlab [Mat07]. Lorsque la pente $df(u_0)/du_0$ est élevée, la dynamique des coefficients l'est aussi. [AFCS08] réduit celle-ci en divisant les coefficients a par un facteur 2^w. Les coefficients a et b des $2^{l+1} \times (b_{u_0} - 1)$ polynômes sont codés en virgule fixe sur (16,2,14) et (16,4,12).

Le nombre de bits de u_1 est choisi égal à 16 bits (de manière arbitraire). Pour approcher les fonctions $g_0(u_0)$ et $g_2(u_0)$, un quart de la fonction sinus est décomposé en 1024 segments. Les valeurs sont directement mémorisées suivant le format virgule fixe (16,0,15).

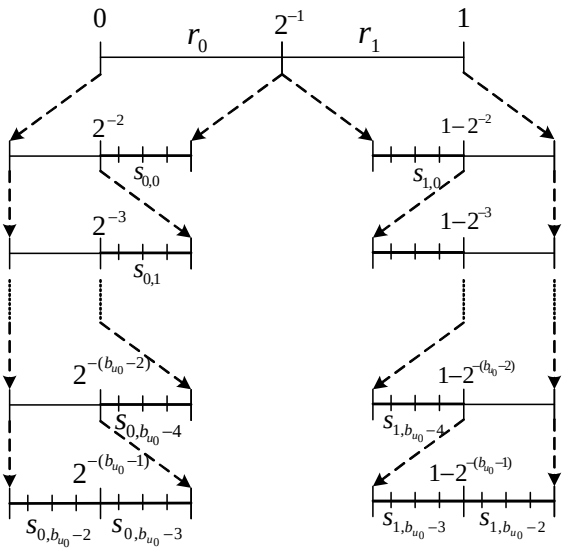

Figure 3-11 – Segmentation hybride [AFCS08].

3.2 Implantation de l'algorithme Box-Muller à partir d'une approximation polynomiale minimax

L'implantation proposée par [AFCS08] est intéressante pour le principe de segmentation utilisé. L'objectif de cette partie est de proposer une implantation de l'algorithme Box-Muller dans laquelle des approximations minimax sont effectuées. De plus l'objectif est d'obtenir un GRNG à très haut débit. La valeur maximale atteignable en queue de gaussienne ($\max_\sigma$) est prise égale à 4.7σ. Une valeur supérieure n'a pas été prise en raison de la surface du circuit engendrée. L'entrée u_0 doit être représentée par 16 bits.

Une approximation polynômiale minimax est utilisée pour approcher la fonction $f(u_0)$. Le cahier des charges du GRNG est de produire des variables aléatoires au minimum codées sur 16 bits. A priori, les v.a. gaussiennes en sortie du générateur sont représentées en virgule fixe suivant le format (b, m, n), avec $b = 16$, $m = 3$ et $n = 12$. Les notations b, m et n désignent respectivement le nombre de bits sur lequel est codé la donnée, la partie entière et la partie fractionnaire. La précision minimale est donc égale à 12 bits.

Étant donné que la qualité du GRNG repose sur l'implantation de la fonction $f(u_0)$, celle-ci doit avoir une précision d'au moins 12 bits corrects. Afin de satisfaire la précision requise, un certain nombre de bits de garde peut être ajouté au sein du GFS. Par conséquent, la largeur des données en sortie du GRNG est peut être supérieure à 16 bits.

3.2-1 Implantation de la fonction $f(u_0)$

La dérivée $df(u_0)/du_0$ donnée par

$$df(u_0)/du_0 = f'(u_0) = -\frac{1}{u_0\sqrt{(-2\log(u_0))}} \tag{3-21}$$

est présentée dans la figure 3-12. Celle-ci montre la pente de la fonction f. Lorsque les valeurs de u_0 sont proches de 0 ou de 1, le coefficient directeur est très élevé.

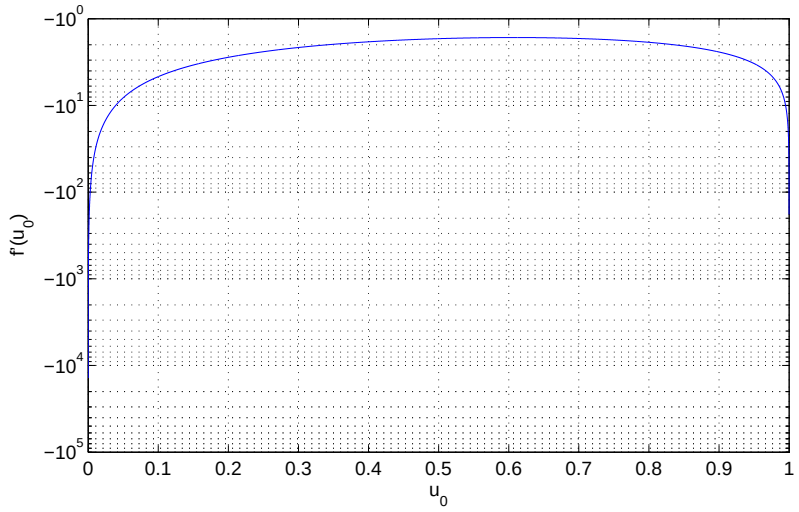

Figure 3-12 – Dérivée de la fonction $f(u_0)$. Celle-ci montre la pente de la fonction f. Lorsque les valeurs de u_0 sont proches de 0 ou de 1, le coefficient directeur est très élevé.

Nous optons pour une segmentation hybride utilisant une décomposition uniforme et logarithmique du domaine de définition de la fonction f. Pour $u_0 \in [2^{-2}, 2^{-1}+2^{-2}+2^{-3}] = [0.25, 0.875]$, le domaine est divisé en segments de tailles uniformes. Ceci en raison de la quasi-linéarité de la fonction. En dehors de cet intervalle, une segmentation logarithmique est réalisée. La suite de cette section détaille l'approximation dans les intervalles.

A. $u_0 \in [0.25, 0.875[$
La fonction représentée dans la figure 3-13 est approchée par des polynômes minimax. Le domaine de la fonction est segmenté en intervalles uniformes. Le nombre de segments minimal permettant de satisfaire le critère de précision est recherché pour des polynômes minimax de degrés 1, 2 et 3. La précision minimale obtenue par les polynômes est résumée dans les tableaux 3-8, 3-9 et 3-10. Ainsi, 32, 8 et 4 sous-segments uniformes sont nécessaires pour approcher la fonction f à partir de polynômes de degrés 1, 2 et 3.

Les coefficients des polynômes sont ensuite représentés en virgule fixe. Le tableau 3-11 résume la représentation des coefficients des polynômes de degrés 1, 2 et 3 permettant de satisfaire les 12 bits de précision. De même, le coût mémoire (en bits) engendré par la mémorisation des coefficients est présenté dans le tableau. Deux configurations sont

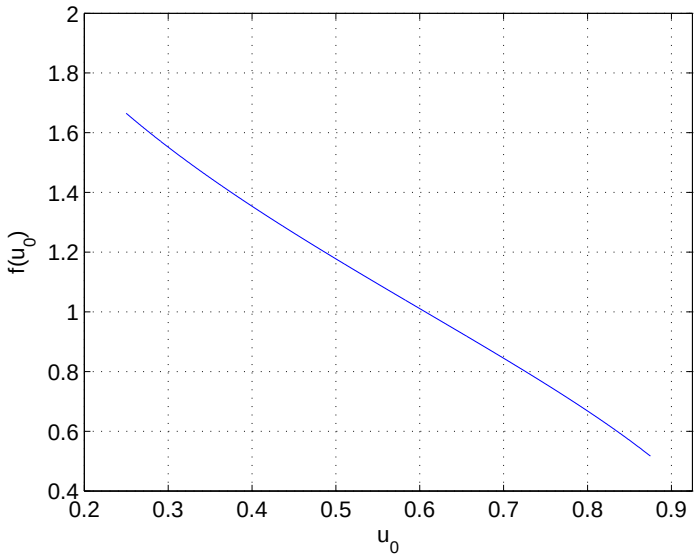

Figure 3-13 – Fonction $f(u_0)$ pour $u_0 \in [0.25, 0.875[$.

Nombre d'intervalles	1	2	4	8	16	**32**
Précision minimale	5.64	6.18	7.52	9.15	10.91	**12.74**

Tableau 3-8 – Précision minimale de $p^\star$ pour $u_0 \in [0.25, 0.875[$ lorsque le polynôme est de degré 1.

analysées : sans et avec réduction de l'intervalle. La réduction de l'intervalle est réalisée en soustrayant la valeur de u_0 à la borne inférieure du segment actif. Ceci permet de réduire la dynamique des coefficients du polynôme et du chemin de données. Cependant, ici, lors de l'utilisation de la réduction sur certains intervalles, le nombre de bits de la partie fractionnaire augmente de un ou deux bits. Par conséquent, le coût mémoire avec réduction est parfois légèrement supérieur à celui obtenu sans réduction.

Les résultats montrent que les polynômes sont représentables sur une largeur de bits peu élevée. De plus, le coût mémoire nécessaire à la mémorisation des coefficients en utilisant un polynôme de degré 2 ou 3 est plus intéressant d'un point de vue des ressources mémoires utilisées. Pour un polynôme de degré 2, une réduction de l'intervalle est utile. Ceci n'est pas le cas pour le polynôme de degré 3.

Nombre d'intervalles	1	2	4	8
Précision minimale	6.0	8.49	10.61	**12.96**

Tableau 3-9 – Précision minimale de $p^\star$ pour $u_0 \in [0.25, 0.875[$ lorsque le polynôme est de degré 2.

Nombre d'intervalles	1	2	**4**
Précision minimale	9.82	11.15	**13.69**

Tableau 3-10 – Précision minimale de $p^\star$ pour $u_0 \in [0.25, 0.875]$ lorsque le polynôme est de degré 3.

Degré	Représentation virgule fixe		Coût mémoire (bits)	
	Sans réduction	Avec réduction	Sans réduction	Avec réduction
1	$p_1 = (14, 2, 11)$	$p_1 = (15, 2, 12)$	896	928
$(p_1 u_0 + p_0)$	$p_0 = (14, 2, 11)$	$p_0 = (14, 1, 12)$		
	$p_2 = (14, 2, 11)$	$p_2 = (14, 2, 11)$		
2	$p_1 = (14, 2, 11)$	$p_1 = (14, 2, 11)$	336	328
$(p_2 u_0^2 + p_1 u_0 + p_0)$	$p_0 = (14, 2, 11)$	$p_0 = (13, 1, 11)$		
	$p_3 = (14, 3, 10)$	$p_3 = (16, 3, 12)$		
3	$p_2 = (15, 4, 10)$	$p_2 = (15, 2, 12)$	228	240
$(p_3 u_0^3 + p_2 u_0^2$	$p_1 = (15, 4, 10)$	$p_1 = (15, 2, 12)$		
$+p_1 u_0 + p_0)$	$p_0 = (13, 2, 10)$	$p_0 = (14, 1, 12)$		

Tableau 3-11 – Représentation en virgule fixe des coefficients et coût mémoire en bits avec et sans réduction de l'intervalle.

B. $u_0 \in [2^{-16}, 2^{-2}[$

La fonction $f(u_0)$ pour $u_0 \in [2^{-16}, 2^{-2}[$ est représentée dans la figure 3-14.

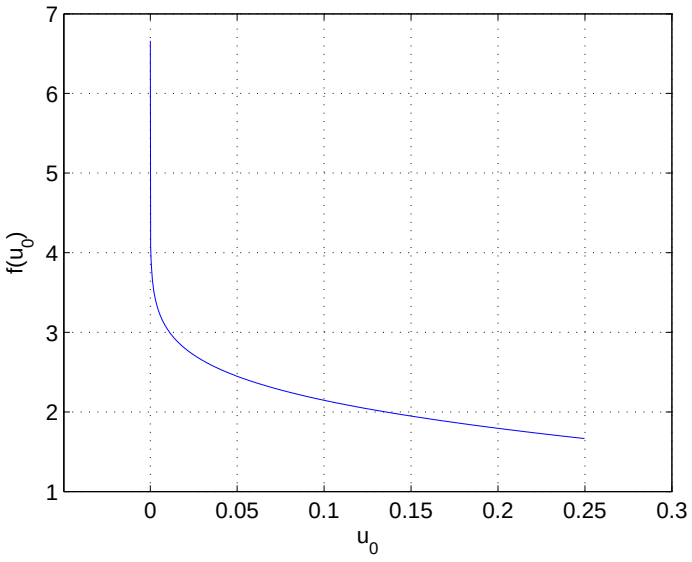

Figure 3-14 – Fonction $f(u_0)$ pour $u_0 \in [2^{-32}, 2^{-2}[$.

La pente élevée de la fonction pousse à utiliser une segmentation logarithmique vers le bas. Par exemple, pour un nombre de segments égal à 3, la fonction $f(u_0)$ est approchée sur trois intervalles : $[2^{-16}, 2^{-4}[$, $[2^{-4}, 2^{-3}[$ et $[2^{-3}, 2^{-2}[$.

Le domaine de f est segmenté en 13 intervalles. La figure 3-15 représente la précision obtenue par des polynômes de degrés 2 et 3. Comme l'indique la figure, un polynôme de degré 2 est insuffisant pour atteindre une précision de 12 bits. Par conséquent, des polynômes de degré 3 sont utilisés par la suite.

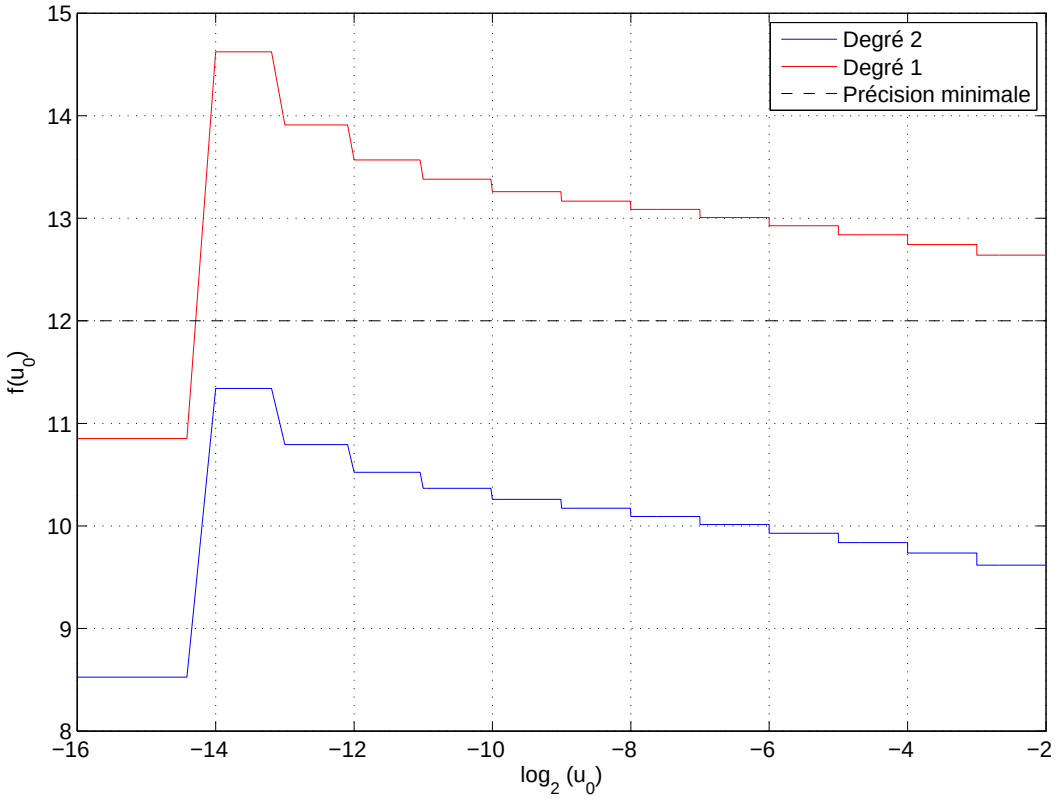

Figure 3-15 – Précision obtenue lors l'approximation de $f(u_0)$ pour $u_0 \in [2^{-16}, 2^{-2}[$ avec des polynômes de degrés 2 et 3.

Un grand nombre de segments est nécessaire pour satisfaire la contrainte de précision de 12 bits corrects. Cependant, l'efficacité d'une approximation polynômiale par rapport à une mémorisation directe doit êtré évaluée. La figure 3-16 présente le coût mémoire total (en bits) lorsque la fonction $f(u_0)$ est approchée pour $u_0 \in [2^j, 2^{-2}[$ et mémorisée pour $u_0 \in [2^{-16}, 2^j[$, avec $j \in \{-14, \ldots, -3\}$. Les deux courbes correspondent à une approximation avec et sans réduction de l'intervalle. Le coût mémoire des deux méthodes est similaire. Celui-ci est minimal pour $j = -12$ et égal à 1450 bits (1480 bits sans réduction de l'inter-

Degré du polynôme	Représentation en virgule fixe	
	Sans réduction intervalle	Avec réduction intervalle
3 $(p_3u_0^3 + p_2u_0^2 + p_1u_0 + p_0)$	$p_3 = (45, 31, 13)$	$p_3 = (45, 31, 13)$
	$p_2 = (36, 22, 13)$	$p_2 = (35, 21, 13)$
	$p_1 = (26, 12, 13)$	$p_1 = (24, 10, 13)$
	$p_0 = (17, 3, 13)$	$p_0 = (17, 3, 13)$

Tableau 3-12 – Représentation en virgule fixe des coefficients.

valle). Par conséquent, la fonction $f(u_0)$ est approchée pour $u_0 \in [2^{-12}, 2^{-2}[$ (soit 16368 valeurs possibles) et mémorisée pour $u_0 \in [2^{-16}, 2^{-12}[$ (soit 15 valeurs possibles).

Le tableau 3-12 résume la représentation en virgule fixe des coefficients avec et sans réduction de l'intervalle. Les largeurs des coefficients sont peu élevées. Cependant la dynamique du coefficient p_3 est plus importante que celle des autres coefficients. Ceci est engendré par la présence de l'asymptote verticale pour des valeurs faibles de u_0.

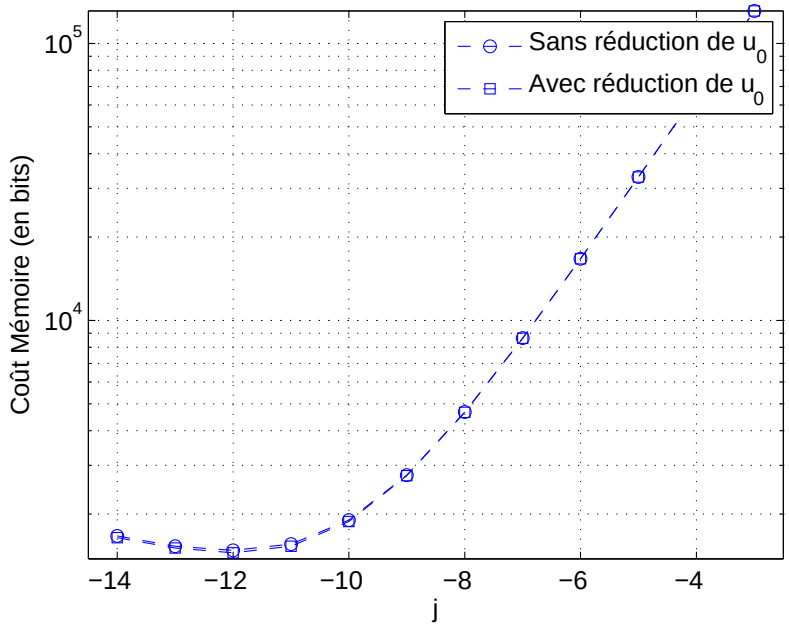

Figure 3-16 – Coût mémoire total (en bits) lorsque la fonction $f(u_0)$ est approcher pour $u_0 \in [2^j, 2^{-2}[$ et mémorisée pour $u_0 \in [2^{-16}, 2^j[$ pour $j \in \{-14, \ldots, -3\}$.

C. $u_0 \in [0.875, 1 - 2^{-16}]$

La fonction à approcher $f(u_0)$ pour $u_0 \in [0.875; 1[$ est présentée dans la figure 3-17. Étant

donné que pour l'intervalle $u_0 \in [2^{-12}, 2^{-2}[$ un polynôme d'ordre trois est requis, un polynôme du même ordre est utilisé ici.

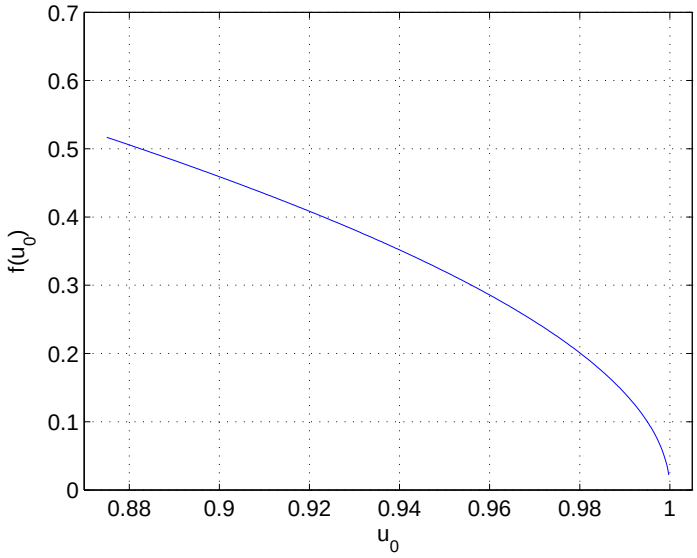

Figure 3-17 – Fonction $f(u_0)$ pour $u_0 \in [0.875; 1[$.

Le domaine de la fonction est segmenté en intervalles de tailles logarithmiques. La figure 3-18 présente le schéma de segmentation proposé. Celui-ci a été choisi afin de minimiser le nombre de segments tout en satisfaisant les 12 bits de précision requis. Le schéma de segmentation est dit non uniforme car, à la différence de l'approximation de $f(u_0)$ pour $u_0 \in [2^{-16}, 2^{-2}[$, un segment n'est pas adressé à chaque bit de u_0. Ceci est rendu possible ici grâce au faible nombre de segments requis. Le domaine de la fonction est décomposé en six segments différents.

Après recherche des représentations des coefficients en virgule fixe, la figure 3-19 présente le coût mémoire engendré par une approximation réalisée sur les j premiers intervalles. Dans ces conditions, les valeurs appartenant aux intervalles restants sont directement tabulées. Pour $j = 5$, le coût mémoire est minimisé et est égal à 583 bits avec réduction de l'intervalle (868 bits sans réduction de l'intervalle). Par conséquent, sur l'intervalle étudié, la fonction $f(u_0)$ est approchée pour $u_0 \in [0.875, 9.98]$ et tabulée pour $u_0 \in]9.98, 1[$ (16 valeurs).

La représentation des coefficients en virgule fixe est donnée dans le tableau 3-13 sans et avec réduction de l'intervalle. La largeur des coefficients p_2, p_1 et p_0 est très largement inférieure avec réduction de l'intervalle et montre l'intérêt de la méthode.

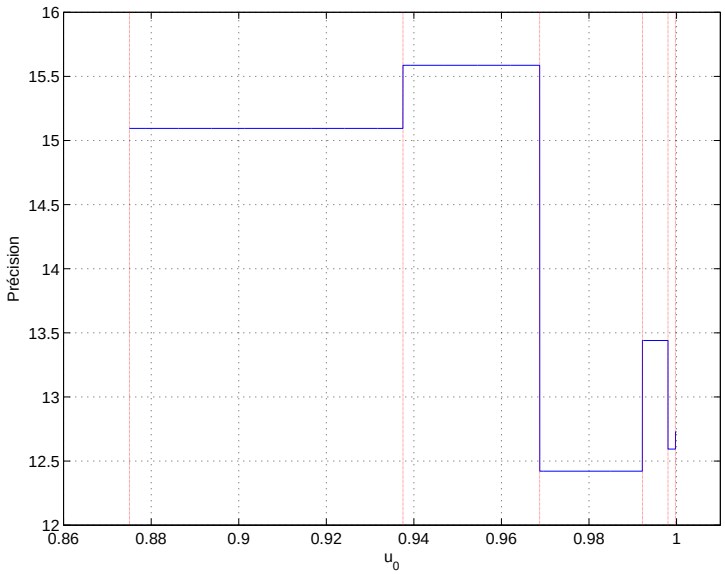

Figure 3-18 – Schéma de segmentation et précision de l'approximation de $f(u_0)$ pour $u_0 \in [0.875, 1[$.

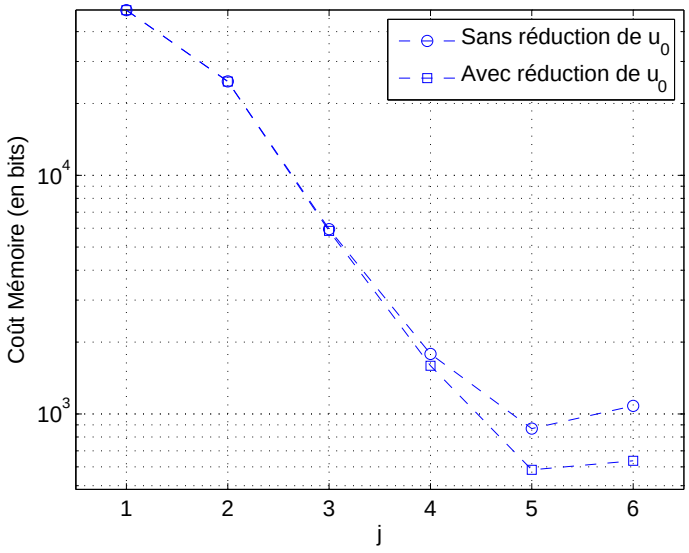

Figure 3-19 – Coût mémoire total (en bits) engendré par une approximation réalisée sur les j premiers intervalles.

Le tableau 3-14 résume le coût mémoire total (en bits) engendré par l'approximation polynômiale de la fonction $f(u_0)$ pour $u_0 \in [2^{-16}, 1[$ sans et avec réduction de l'intervalle. De plus, le coût d'une méthode utilisant une mémorisation en tables des valeurs de $f(u_0)$ est aussi présenté. L'approximation la plus efficace en termes de mémoire utilisée est celle utilisant une réduction de l'intervalle. Au total, 2057 bits de mémoire sont nécessaires à la

Degré du polynôme	Représentation en virgule fixe	
	Sans réduction intervalle	Avec réduction intervalle
3 $(p_3u_0^3 + p_2u_0^2 + p_1u_0 + p_0)$	$p_3 = (36, 22, 13)$	$p_3 = (35, 22, 12)$
	$p_2 = (38, 24, 13)$	$p_2 = (25, 12, 12)$
	$p_1 = (38, 24, 13)$	$p_1 = (18, 5, 12)$
	$p_0 = (36, 22, 13)$	$p_0 = (13, 0, 12)$

Tableau 3-13 – Représentation en virgule fixe des coefficients.

	Approximation polynômiale		Mémorisation
	Sans réduction	Avec réduction	Directe
Coût mémoire (bits)	2576	**2057**	983 025

Tableau 3-14 – Coût mémoire (en bits) engendré par l'approximation polynômiale et par une mémorisation directe des valeurs de $f(u_0)$ sur les cinq segments.

sauvegarde de l'ensemble des coefficients et des valeurs de $f(u_0)$ tabulées.

La taille minimale des données du graphe de calcul de la figure 3-20 est recherchée de manière exhaustive. Comme le montre la figure 3-20, deux bits de garde sont nécessaires lors de la dernière multiplication pour satisfaire la contrainte de précision.

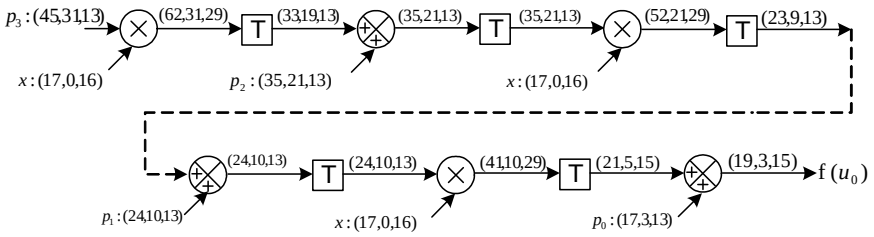

Figure 3-20 – Graphe flot de signal réalisant le calcul de $f(u_0)$.

3.2-2 Implantation de la fonction $g_0(u_1)$

La fonction

$$g_0(u_1) = sin(2\pi u_1) \tag{3-22}$$

est tracée dans la figure 3-21 pour $u_1 \in [0, 1]$. L'implantation de la fonction peut-être simplifiée en utilisant les propriétés de symétrie de la fonction sinus. Ainsi, seul $g_0(u_1)$ pour $u_1 \in [0, 0.25]$ doit être calculé. De la même manière, pour la fonction $f(u_0)$, une approximation polynômiale minimax est ici proposée.

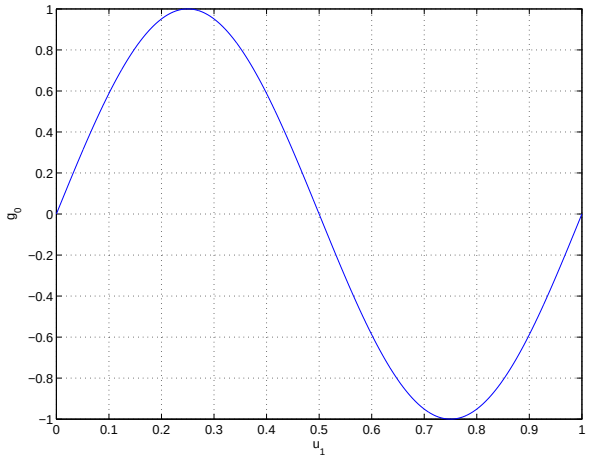

Figure 3-21 – Fonction $g_0(u_1)$.

Degré	Nombre de segments	Précision minimale	Largeur en bits partie fractionnaire	Coût mémoire (en bits)
2	8	14.63	12	363
3	2	13.1	12	124

Tableau 3-15 – Nombre de segments requis, nombre de bits consacrés à la partie fractionnaire des coefficients, précision minimale de l'approximation et coût mémoire engendré par des polynômes de degrés 2 et 3.

Une décomposition uniforme du domaine de u_1 est effectuée avec réduction de l'intervalle. Le tableau 3-15 présente le nombre de segments requis, le nombre de bits consacré à la partie fractionnaire des coefficients, la précision minimale de l'approximation et le coût mémoire engendré par des polynômes de degrés 2 et 3. L'utilisation d'un polynôme de degré deux impose de segmenter le domaine u_1 en 8 intervalles. Ainsi, une mémoire de 363 bits est requise pour tabuler les coefficients. En comparaison, l'utilisation d'un polynôme de degré 3 entraîne la segmentation du domaine en 2 intervalles. Une mémoire de 124 bits est ainsi nécessaire. D'un point de vue coût mémoire, il est préférable d'utiliser un polynôme de degré 3.

La taille minimale des données du graphe flot de calcul présenté dans la figure 3-22 est ensuite recherchée de manière exhaustive. Comme le montre la figure 3-22, un bit de garde est nécessaire lors de la dernière multiplication pour satisfaire la contrainte de précision. La figure présente également la représentation en virgule fixe des coefficients des polynômes.

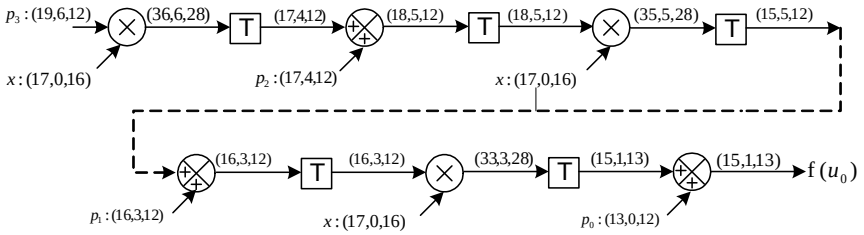

Figure 3-22 – Graphe flot de signal réalisant le calcul de $g_0(u_1)$.

Fonction	$f(u_0)$	$g_0(u_1)$
Nombre total de LUT	480 (1%)	132 (1%)
Fréquence maximale	119 MHz	289 MHz
BRAM	0 (0%)	0 (0%)
DSP48	7 (2%)	3
Largeur de sortie (bits)	19	16
Débit (Gb/s)	2.261	4.6245

Tableau 3-16 – Résultats d'implantation pour une cible FPGA Xilinx Virtex 5 SX50. Présentation du nombre total de LUT, de la fréquence maximale, du nombre de BRAM et de DSP48. De plus, la largeur du mot en sortie et le débit du générateur sont présentés.

3.2-3 Résultats d'implantation des fonctions $f(u_0)$ et $g_0(u_1)$

Les circuits réalisant le calcul des fonctions $f(u_0)$ et $g_0(u_1)$ sont synthétisés et placés-routés au sein d'un FPGA Xilinx Virtex 5 SX50. Le tableau 3-16 résume les coûts et performances des deux fonctions. Comme indiqué, l'implantation des fonctions ne nécessite pas d'utiliser des BRAM. Les fonctions $f(u_0)$ et $g_0(u_1)$ utilisent resepectivement 480 LUT et 132 LUT. Ceci représente moins d'un pourcent des capacités du circuit. La fréquence maximale de $f(u_0)$ est de 119 MHz. La fréquence maximale de $g_0(u_1)$ est égale à 289 MHz. La différence entre ces deux fréquences s'explique par la taille supérieure des multiplieurs utilisés dans le graphe du calcul de $f(u_0)$.

Un canal de Rayleigh se limitant à l'utilisation de la fonction $f(u_0)$, les performances obtenues sont similaires à celles résumées dans le tableau 3-16.

3.2-4 Résultats d'implantation de l'algorithme Box-Muller

Afin de maximiser le débit obtenu par l'algorithme Box-Muller, deux blocs $g_0(u_1)$ et un bloc $f(u_0)$ sont utilisés. Les calculs du sinus et cosinus sont effectués en parallèle. Ainsi, le circuit est efficacement pipelinable. Le tableau 3-17 résume les performances obtenues par l'architecture développée après placement et routage au sein d'un FPGA Xilinx Virtex 5 SX50. La fréquence maximale du circuit est égale à 119 Mhz. Le nombre de DSP48 requis

Fonction	Box-Muller
Nombre total de LUT	777 (2%)
Fréquence maximale	119 MHz
BRAM	0 (0%)
DSP48	15 (5%)
Largeur de sortie (bits)	19
Débit (Gb/s)	4.522

Tableau 3-17 – Résultats d'implantation pour une cible FPGA Xilinx Virtex 5 SX50.

FPGA ciblé	Xilinx Virtex-II XC2V4000-6					
	[Xil02]	[LLVC04]	[ABS05]	[Lee06]	[AFCS08]	Méthode présentée
Cellules logiques	480	2514	702	1528	534	653
Blocs RAM	5	2	5	3	2	0
MULT 18x18	5	8	1	12	3	17
fréquence max.	245	133	165	233	248	92
Débit (Mbits/s)	245	133	165	466	496	3496
$\sigma_{\max}$	4.8σ	6.7σ	5σ	8.2σ	$6.66\ \sigma$	4.7σ

Tableau 3-18 – Comparaison de la performance de l'architecture présentée avec les GRNG utilisant la méthode BM.

est égal à 15. Le débit total du circuit est de 4.522 Gb/s.

Afin de comparer les performances obtenues par le circuit avec les GRNG existants, celui-ci est implanté au sein d'un FPGA Xilinx Virtex-II XC2V4000-6. Les performances du circuit après placement et routage sont résumées dans le tableau 3-18. En comparaison, notre circuit est plus coûteux en termes de LUT et de multiplieurs utilisés. Ceci est dû au fait qu'aucun bloc RAM n'est requis par notre architecture. Les coefficients sont directement tabulés au sein de LUT, alors que les autres GRNG utilisent des blocs RAM. La fréquence de notre circuit est plus faible aussi que les autres GRNG. Cependant, le débit obtenu par notre circuit est très largement supérieur aux autres GRNG. Ceci s'explique par la possibilité de pipeliner notre calcul de manière très efficace ce qui permet de réaliser un nouveau calcul tous les cycles d'horloge.

3.2-5 Validation de la qualité de l'algorithme Box-Muller implanté

Le générateur Box-Muller est implanté et validé au sein du FPGA Xilinx Virtex 5 SX50. Les données issues de celui-ci sont mémorisées. La figure 3-23 représente 10^4 couples de variables aléatoires générées par le GRNG. La distribution des couples semble suivre une loi normale.

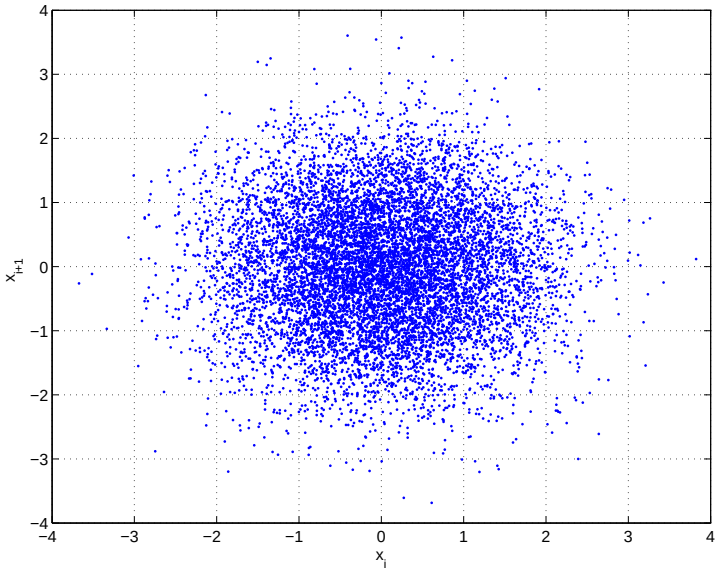

Figure 3-23 – Représentation de 10^4 couples de variables aléatoires générées par le GRNG.

La figure 3-24 présente l'histogramme obtenu à partir de 2×10^6 réalisations de la variable aléatoire gaussienne X. Celle-ci est obtenue en concaténant les variables aléatoires X_1 et X_2. L'histogramme suit une loi normale. Cependant, la variable aléatoire X doit être évaluée plus finement à partir de tests statistiques.

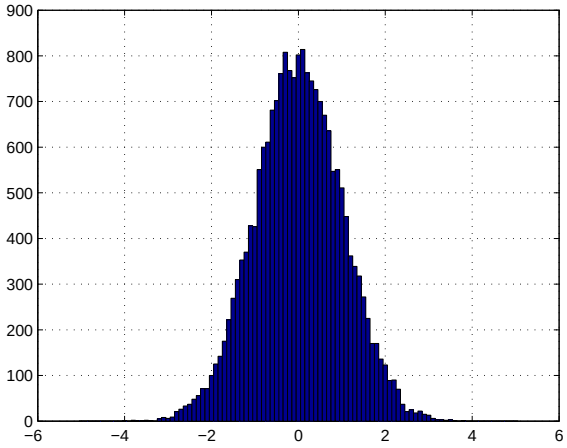

Figure 3-24 – Histogramme de la variable aléatoire gaussienne.

Un test du χ^2 est réalisé sur X. Le test compare l'histogramme théorique d'une loi normale et celui observé. Pour une probabilité de première espèce $\alpha = 5\%$, le test déclare que la variable aléatoire X suit une loi normale centrée réduite. Ensuite, le test d'Anderson-Darling est effectué sur X. Ce test est une version modifié du test de Kolmogorov-Smirnov (KS) et permet d'évaluer plus finement la distribution de la queue de la gaussienne. Pour une probabilité de première espèce $\alpha = 5\%$, le test d'Anderson-Darling est réussi. La variable aléatoire X est considérée suivre une loi normale centrée réduite.

3.3 Conclusions

Après avoir présenté les techniques de GRNG existantes, la méthode Box-Muller a été sélectionnée. Celle-ci a été choisie pour l'efficacité de son implantation, la possibilité d'utiliser une partie de l'algorithme afin de générer des variables aléatoires de Rayleigh et la qualité des échantillons générés.

Une implantation de l'algorithme basée sur l'utilisation d'approximation polynomiale est présentée. Les fonctions sont segmentées afin de minimiser le coût mémoire engendré par l'approximation. Des polynômes de degré trois sont utilisés pour approcher les fonctions $f(u_0)$ et $g_0(u_1)$. Le circuit placé et routé occupe un nombre de LUT et multiplieurs supérieur aux GRNG existants. Cependant, à la différence des autres contributions, le circuit ne contient aucun bloc RAM. Par ailleurs, le circuit étant facilement pipelinable, il est possible d'obtenir deux variables aléatoires codées sur 19 bits à chaque cycle d'horloge. Au sein d'un FPGA Xilinx Virtex 5 SX50, le GRNG possède un débit de 4.522 Gb/s.

Dans le cas d'un générateur de variables aléatoires de Rayleigh, le circuit réalisant la fonction $f(u_0)$ peut être utilisé. Au sein d'un FPGA Xilinx Virtex 5 SX50, le générateur possède un débit de 2.261 Gb/s.

Dans cette section, la valeur de $\sigma_{\max}$ a été bornée à 4.7σ. La suite logique de ce travail serait dans un premier temps d'optimiser le GFS utilisé par la fonction $f(u_0)$ afin d'augmenter le débit du GRNG puis d'étudier le coût matériel engendré par l'utilisation d'une valeur $\sigma_{\max}$ supérieure à 4.7σ.

4 Conclusion

La première sous-section a présenté l'implantation d'un générateur hybride au sein d'un circuit reconfigurable. Le générateur, développé conjointement avec l'entreprise spécialisée en cryptographie, innove par les mécanismes de test en ligne mis en œuvre pour assurer la qualité du TRNG. L'analyse temps réel de la qualité du générateur est réalisée par l'utilisation des tests matériels du FIPS 140-2. Le coût négligeable engendré par l'implantation des tests au sein d'un circuit reconfigurable rend leur emploi très intéressant. Le composant permet d'assurer au PRNG, une séquence d'initialisation de très bonne qualité.

Dans certaines applications, une implantation au sein d'un circuit dédié (i.e. ASIC) est souvent requise. Le chapitre II a permis de sélectionner un TRNG au sein d'une plateforme reconfigurable. En technologie ASIC, le TRNG à choisir peut-être différent. La suite du chapitre présente l'implantation de deux générateurs uniformes au sein de circuits ASIC. Le premier circuit est composé du TRNG de Schellekens et al. [SPV06], du FIPS 140-2 et d'un automate cellulaire. Les premiers résultats de validation du circuit sont présentés. Le débit maximal du TRNG est de 1.25 Mb/s. Au delà, la batterie FIPS 140-2 détecte des défaillances non négligeables en sortie du générateur. Le fonctionnement correct du circuit nous a permis d'entrevoir la possibilité de développer un second circuit ASIC. Celui-ci, plus intéressant, va permettre d'étudier et de comparer la qualité de quatre TRNG en fonction des conditions externes appliquées sur le circuit. Les batteries FIPS 140-2 et AIS 31 sont implantées en matériel. Le FIPS 140-2 validera, en temps réel la qualité du générateur. L'AIS 31 permettra une analyse plus poussée de la qualité des générateurs. Après sélection et optimisation du débit des TRNG, la sortie aléatoire du générateur choisi servira à initialiser un AES-128. En sortie du PRNG, le débit maximal atteint est égal à 2.57 Gb/s.

Il est parfois nécessaire de générer des nombres aléatoires suivant une distribution de probabilité différente d'une loi uniforme. En communication numérique, des lois normales et de Rayleigh sont très fréquemment utilisées. Par conséquent, un des objectifs du second circuit ASIC est de pouvoir générer de telles lois à très haut débit. À partir de l'étude des solutions existantes pour générer des variables aléatoires gaussiennes, un algorithme est choisi pour la qualité des échantillons générés et du débit atteignable par son implantation, l'algorithme Box-Muller. De plus, à partir d'une fraction de l'algorithme Box-Muller, des variables aléatoires suivant une loi de Rayleigh peuvent être générées. Une implantation de l'algorithme basée sur l'utilisation d'approximations polynomiales est présentée. La surface du circuit, légèrement supérieure aux contributions présentées dans la littérature en termes de LUT et de multiplieurs utilisés, présente l'avantage de ne nécessiter aucun bloc RAM. Le débit maximal en sortie du GRNG et du générateur de Rayleigh sont respectivement de 4.522 Gb/s et de 2.26 Gb/s. Les débits très élevés et la faible surface des circuits permettent de valider les objectifs de départ.

Chapitre

4

Recherche d'un générateur pseudo-aléatoire optimisé selon un compromis qualité/coût matériel

Dans certaines applications, par exemple en simulation numérique, les RNG sont sélectionnés pour leur débit et leurs propriétés statistiques. La robustesse et l'imprédictibilité du générateur ne sont pas aussi primordiales que pour des applications cryptographiques. Le débit maximal atteint par les générateurs vraiment aléatoires n'est pas suffisamment élevé pour ces applications. La seule alternative est d'utiliser un générateur pseudo-aléatoire. Un grand nombre de PRNG existent dans la littérature. Cependant, peu d'entre eux présentent une implantation très efficace au sein de circuits FPGA.

Ce chapitre est une étude annexe à la thèse consistant à rechercher un générateur pseudo-aléatoire optimisé selon un compromis qualité/coût matériel. Cette partie présente les résultats préliminaires de l'étude.

1 Intérêt des Automates Cellulaires dans les architectures FPGA

Les FPGA sont construits autour d'une matrice de cellules logiques reliées entre elles à travers des ressources de routages programmables. Les cellules logiques sont composées par une ou plusieurs tables de vérité à plusieurs entrées (4 à 8 dépendamment du FPGA utilisé), de ressources combinatoires et de bascules D.

Un grand nombre de générateurs pseudo-aléatoires existent dans la littérature. Pierre L'Écuyer [LS07] a récemment étudié les propriétés statistiques de 102 générateurs pseudo-aléatoires. L'analyse a été effectuée en utilisant les trois batteries de tests statistiques TestU01. Parmi les générateurs testés, seuls 22 générateurs valident l'ensemble des tests statistiques. Cependant, la grande majorité des PRNG analysés est conçue à des fins logicielles et peu adaptée à un contexte matériel. Ceux-ci nécessitent souvent des opérateurs complexes (multiplication modulaire, division) et ne profitent pas de la structure parallèle d'un FPGA. En pratique, les générateurs les plus efficaces au sein de circuits FPGA et régulièrement utilisés dans le domaine du test sont les LFSR et les automates cellulaires

(*Cellular Automata*) (CA). Les LFSR sont basés sur des opérations modulo 2. Cependant, ces générateurs échouent fortement aux tests statistiques TestU01 [LS07].

Un automate cellulaire est une structure fortement parallèle et très efficace en vue d'une implantation matérielle. Le schéma de principe d'un CA est donné à la figure 4-1 et est basé sur la connexion locale de N cellules identiques. Dans le schéma, la sortie O1 est reliée à l'entrée I2 et la sortie O2 est reliée à l'entrée I1 de la dernière cellule. Un CA est un tableau de cellules dans lequel chaque cellule possède des connexions avec ses cellules voisines. Une cellule est une machine à états finies. Dans le cas binaire, une cellule peut produire un "0" ou un "1" logique. La règle d'un CA définit l'état futur d'une cellule en fonction de l'état des cellules voisines. En choisissant une règle adéquate, un CA est capable de générer une séquence fortement aléatoire. La qualité d'un CA dépend du vecteur d'initialisation ainsi que du nombre de voisins d'une cellule. La règle d'un CA est décrite par une table de vérité. Dans un FPGA, celle-ci est implantée dans une *Look-Up Table* (LUT). Ainsi, si une LUT possède 4 entrées, chaque cellule peut prendre 16 états différents. La sortie de chaque cellule est une bascule.

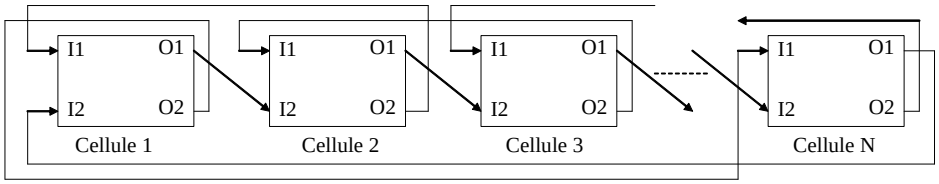

Figure 4-1 – Schéma de principe d'un automate cellulaire [Abd03]. Un automate est basé sur la connexion locale de N cellules identiques.

La règle 30 des automates cellulaires constitue une base historique dans le domaine [Wol86]. Elle correspond à un automate cellulaire mono-dimensionnel à trois voisins. L'état futur d'une cellule est déterminé à partir de l'état de trois cellules voisines. Une cellule a par conséquent $2^3 = 8$ évolutions possibles. La règle 30 est donnée à la figure 4-2 tandis qu'un exemple d'évolution de l'état des cellules est montré à la figure 4-3.

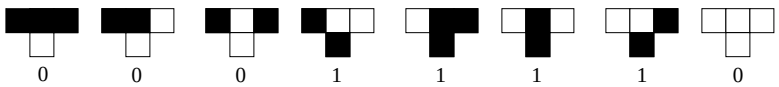

Figure 4-2 – Règle 30 [Wol02].

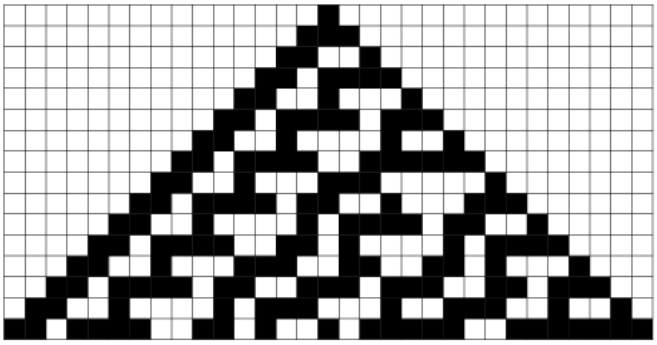

Figure 4-3 – Évolution de l'état des cellules à partir de la règle 30, [Wol02]. L'évolution est basée sur l'initialisation de la cellule centrale à "1" et le reste des cellules à "0". Une ligne correspond à un cycle du générateur.

[GLM05] a comparé la qualité de plusieurs automates cellulaires utilisant la règle 30. Chaque générateur a un nombre de cellules différent. [GLM05] montre que l'utilisation de CA possédant un faible nombre de cellules n'est pas un choix judicieux pour générer une séquence aléatoire de qualité.

[STCS02] a implanté un générateur de nombres aléatoires de 64 bits basé sur l'utilisation d'un CA à quatre voisins. Le générateur génère une séquence aléatoire de 64 bits à une fréquence allant de 214 MHz à 230 MHz. [STCS02] a implanté l'automate cellulaire en pensant d'abord à l'implantation matérielle. En partant du fait qu'une LUT possède 4 entrées, une cellule à 4 voisins est utilisée. [Mar02] a montré l'intérêt d'utiliser une combinaison de plusieurs automates cellulaires afin d'améliorer le caractère aléatoire du générateur.

[Abd03] a implanté un CA mono-dimensionnel. La période de la séquence générée varie de 2^{10} si le générateur est initialisé avec seulement des "1", à 2^{28} si le générateur est initialisé aléatoirement. Ce générateur fonctionne à une fréquence de 50 MHz.

D'après [TLL03], l'utilisation de 30 automates cellulaires occupe 2.1 fois plus de surface qu'un LFSR de 30 bits et permet d'obtenir un meilleur débit et une meilleure séquence aléatoire. D'après [STCS02], la règle 30 (CA30) des automates cellulaires représente la meilleure représentation des automates cellulaires mono-dimensionnels.

De plus, le schéma de connexion d'un CA peut être plus ou moins complexe. Les figures 4-4, 4-5 et 4-6 présentent respectivement un CA mono-dimensionnel (noté CA1D), bi-dimensionnel (noté CA2D) et tri-dimensionnel (noté CA3D) de taille 64 (nombre total de cellules). Ceux-ci sont dit périodiques car les cellules extrêmes (premières et dernières cellules) sont connectées entres elles. Ces CA s'opposent aux automates dans lesquels les entrées de les cellules sont fixées à "0" ou "1".

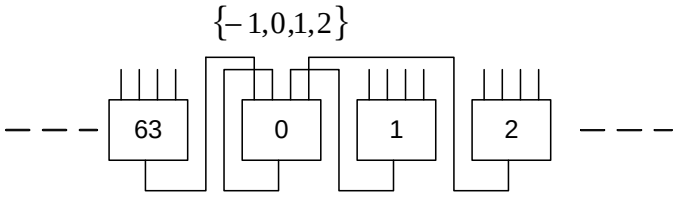

Figure 4-4 – Automate cellulaire mono-dimensionnel. Le schéma de connexion, ici $\{-1, 0, 1, 2\}$, définit les relations entre les cellules.

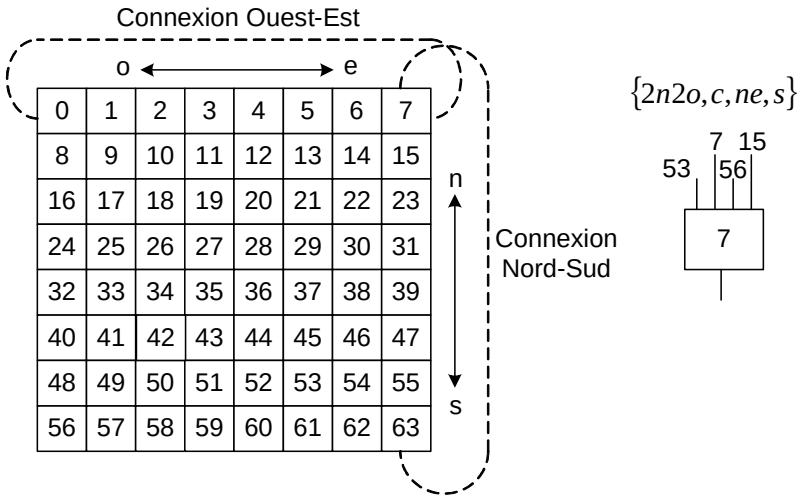

Figure 4-5 – Automate cellulaire bi-dimensionnel. Le schéma de connexion, ici $\{2n2o, c, ne, s\}$, définit les relations entre les cellules. Les notations n, e, s et o désignent les points cardinaux Nord, Est, Sud, Ouest. L'écriture $2n2o$ désigne une cellule située deux positions vers le Nord et deux position vers l'Est. Le c désigne la cellule courante.

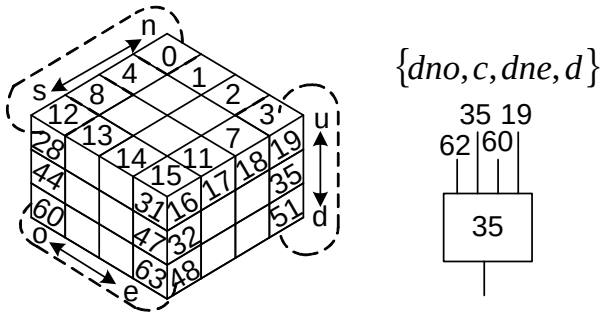

Figure 4-6 – Automate cellulaire tri-dimensionnel. Le schéma de connexion, ici $\{dno, c, dne, d\}$, définit les relations entre les cellules. Les notations n, e, s et o désignent les points cardinaux Nord, Est, Sud, Ouest. Les notations d et u sont utilisées pour indexer la troisième dimension de l'automate. Le c désigne la cellule courante.

De plus, les CA utilisent souvent un écartement spatial (*site spacing*) ou temporel (*time spacing*). Comme le montre la figure 4-7, l'écartement spatial (γ) désigne le nombre de cellules sautées entre deux bits de sortie d'un CA et permet de réduire la corrélation entre les bits de sortie. L'écartement temporel (β) désigne le nombre de cycles d'horloges entre deux générations de nombres pseudo-aléatoires en sortie du CA et réduit la corrélation temporelle.

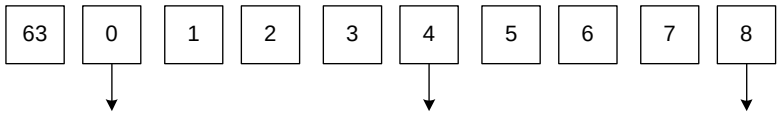

Figure 4-7 – Exemple d'écart spatial avec $\gamma = 3$.

En fonction de ces paramètres, un CA peut produire une séquence aléatoire de très bonne qualité [And01]. Dans ces conditions, les CA sont aussi utilisés en cryptographie. Par exemple, [NKC94] utilisent les CA suivant les règles 102, 150 et 90 pour réaliser un chiffrement par blocs et par chaînes.

Cependant, malgré leur simplicité apparente, prédire le comportement d'un automate à partir d'une analyse mathématique est extrêmement difficile [And01]. Wolfram [Wol02] classifie les CA en quatre classes. La classe 3 spécifie les CA produisant un comportement chaotique imprévisible. L'analyse est réalisée en analysant l'évolution des CA de manière graphique. [HMC89] étudie et compare le comportement statistique de CA suivant les règles 90 et 30 à partir des tests statistiques définis par D. Knuth [Knu97]. L'étude montre qu'en combinant deux règles, l'entropie de la séquence aléatoire produite est supérieure à celle d'un LFSR.

Cependant, en fonction du nombre de voisins d'une cellule, estimer la qualité aléatoire de chaque règle à partir de tests statistiques logiciels est une opération très longue. Si un automate possède N voisins (une cellule à N entrées), le nombre total de règles possibles à analyser est égal à 2^{2^N}. Par conséquent, la procédure de recherche est souvent réalisée par le biais d'algorithmes génétiques [TSP00], [XJMF04], [DVH06]. Si cette méthode permet de converger rapidement vers une règle, mais celle-ci n'est pas forcément optimale.

[STCS02] réalise une recherche exhaustive pour un CA de longueur 64 à quatre voisins. Une mesure d'entropie réalisée sur 16 bits est effectuée sur FPGA. Celle-ci permet de sélectionner un ensemble de 1000 règles (parmi les 2^{16}) ayant une entropie maximale. Ensuite, la règle optimale est choisie par les tests statistiques de NIST [RSNMS01].

2 Recherche de la règle optimale d'un CA à 5 voisins

2.1 Contexte et objectif

Les tables de vérités implantées au sein des circuits FPGA actuels possèdent un nombre d'entrées entre 4 et 8. L'objectif de cette partie est d'étudier l'intérêt d'utiliser un automate cellulaire possédant un nombre d'entrées supérieur à 4. Ainsi, la règle optimale d'un automate cellulaire mono-dimensionnel à 5 entrées, composé de 64 cellules et périodique (i.e. la dernière cellule est connectée à la première) est recherchée. La méthode proposée est une extension de [STCS02].

L'objectif est d'analyser de manière exhaustive les 2^{32} règles possibles. Une analyse via l'utilisation de tests statistiques logiciels est difficilement envisageable. Par conséquent, une méthode efficace pour trouver la règle optimale doit être trouvée.

La méthode proposée est de profiter des performances des FPGA afin d'accélérer la recherche exhaustive. La méthode utilise la batterie matérielle FIPS 140-1 (version antérieure au FIPS 140-2) ainsi qu'une mesure de l'entropie.

2.2 Méthodologie

Comme le présente la figure 4-11, la recherche exhaustive utilise la performance et le fort parallélisme des FPGA. Chaque règle est analysée en temps réel par la batterie matérielle FIPS 140-1. En parallèle, l'entropie de la séquence générée est mesurée. La règle possédant l'entropie la plus élevée et satisfaisant les tests statistiques est conservée.

2.2-1 Génération des règles possibles

Un automate cellulaire à 5 entrées possède 2^{32} règles possibles. En réalité, le nombre de règles peut-être réduit. Pour générer une séquence aléatoire équiprobable, chaque règle doit contenir autant de 0 que de 1 logiques dans sa table de vérité. Par conséquent, le nombre de règles à tester est égal à

$$C_{32}^{16} = 601.09 \times 10^6. \tag{4-1}$$

Un algorithme générant l'ensemble des combinaisons possibles doit être implanté. La solution logicielle proposée par [Kok97] est intéressante en vue d'une implantation matérielle. Celle-ci présente l'avantage de travailler à partir d'opérations binaires très simples. Le tableau 4-1 résume les performances et le coût matériel de l'algorithme implanté dans une cible FPGA Xilinx Virtex 5 SX50. La solution permet d'atteindre une fréquence de fonctionnement très élevée et occupe une surface négligeable (seulement 309 LUT).

FPGA ciblé	Xilinx Virtex 5 XC5VSX50
Nombre total de LUT	309 (0%)
Fréquence maximale (MHz)	217.0 MHz

Tableau 4-1 – Résultat de l'implantation de l'algorithme produisant les C_{32}^{16} combinaisons possibles.

2.2-2 Calcul de l'entropie de l'automate cellulaire

Pendant l'analyse par les tests statistiques, la séquence binaire est décomposée en 2.5×10^3 blocs disjoints de 8 bits. Ceux-ci sont obtenus en utilisant la sortie binaire de la $32^{\text{ème}}$ cellule du CA. L'entropie 8 bits

$$H_8 = -\sum_{i=0}^{2^8-1} p_i \log_2 p_i \tag{4-2}$$

de la séquence aléatoire est calculée, p_i représente la probabilité d'obtenir la $i^{\text{ième}}$ sous-séquence possible. L'entropie maximale est égale à $H_{max} = 8$.

La figure 4-8 présente le principe du calcul de (4-2). Une mémoire accessible par son contenu (CAM) est utilisée afin de mémoriser le nombre d'occurrences des blocs 8 bits. En théorie, la valeur de probabilité p_i appartient à l'ensemble $\{1/2500, 2/2500 \ldots, 2499/2500, 1\}$. Ici, le cas où $p_i = 0$ est évité.

La fonction $p_i \log_2 p_i$, représentée dans la figure 4-9, doit être calculée pour

$$p_i \in \{\frac{1}{2.5 \times 10^3}, \ldots, 1\}.$$

L'erreur maximale autorisée en sortie de (4-2) est égale à 10^{-4}, soit une précision de 13.28 bits corrects. La précision nécessaire en sortie du calcul de $p_i \log_2 p_i$ est donc de 21.28 bits corrects. Afin de réduire le coût mémoire, le calcul de l'entropie est arrêté si le nombre d'occurrences d'un bloc 8 bits est quatre fois supérieur à la valeur idéale ($2.5 \times 10^3/2^8 \approx 9.76$) (lorsque les blocs sont uniformément répartis). Par conséquent, la fonction $p_i \log_2 p_i$ doit être calculée pour $p_i \in \{\frac{1}{2.5\times10^3}, \ldots, \frac{40}{2.5\times10^3}\}$. Celle-ci est représentée dans la figure 4-10.

Le coût mémoire nécessaire à la mémorisation des 40 valeurs possibles de $p_i \log_2 p_i$ est égal à $40 \times 22 = 880$ bits. Les parties entières et fractionnaires des données sont respectivement représentées par 0 et 22 bits.

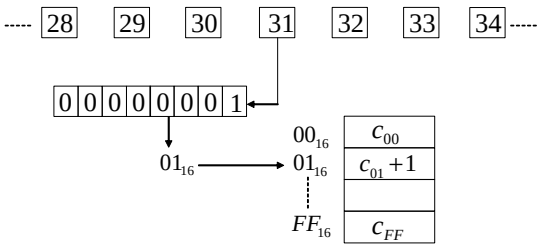

Figure 4-8 – La figure présente le principe du calcul de (4-2). Une mémoire accessible par son contenu (CAM) est utilisée afin de mémoriser le nombre d'occurrences des blocs 8 bits.

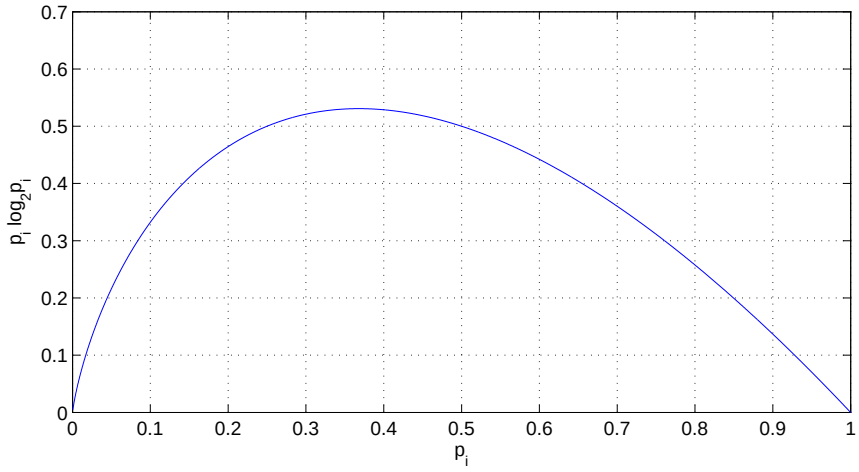

Figure 4-9 – Fonction $p_i \log_2 p_i$ pour $p_i \in \{\frac{1}{2.5\times10^3}, \ldots, 1\}$.

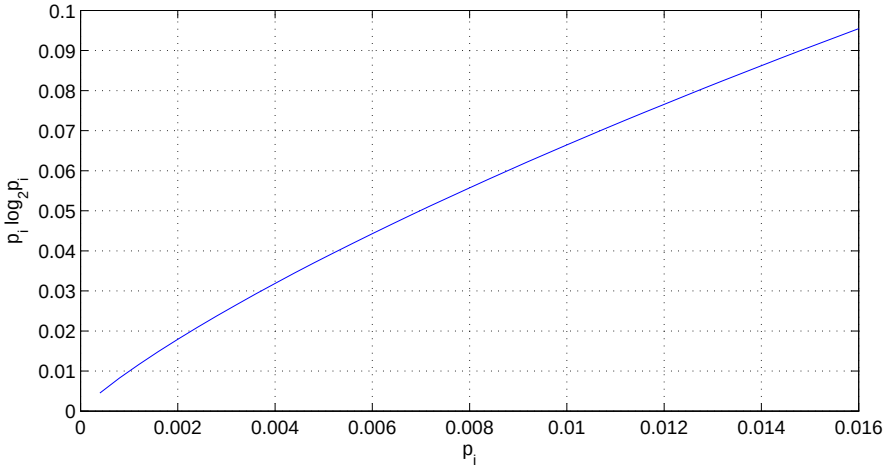

Figure 4-10 – Fonction $p_i \log_2 p_i$ pour $p_i \in \{\frac{1}{2.5\times10^3}, \ldots, \frac{40}{2.5\times10^3}\}$.

Ici, une approximation polynomiale de la fonction serait envisageable afin de réduire le

coût mémoire.

2.3 Architecture de la recherche exhaustive

La règle optimale d'un CA1D à cinq voisins de connexion $\{-2, -1, c, 1, 2\}$ est recherchée. Le schéma de connexion a été choisi afin de minimiser le chemin critique de l'automate.

La figure 4-11 présente l'architecture de la recherche exhaustive. Celle-ci utilise N_b batteries en parallèle. Les automates cellulaires sont implantés à partir de blocs RAM. Il est possible de reconfigurer facilement les règles des CA. L'architecture est implantée dans un FPGA Virtex 5 LX 50. La fréquence du circuit est de 188.83 MHz. La recherche exhaustive est réalisée en 3 heures et 32 minutes.

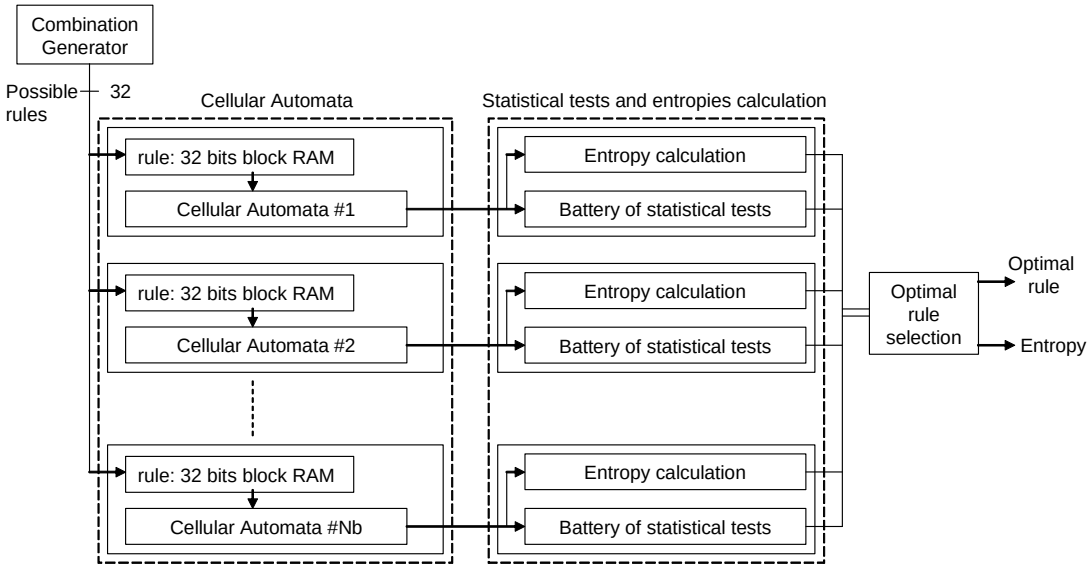

Figure 4-11 – La figure présente l'architecture de la recherche exhaustive. Celle-ci utilise N_b batteries en parallèle. Les automates cellulaires sont implantés à partir de blocs RAM. Il est possible de reconfigurer facilement les règles des CA. L'architecture est implantée dans un FPGA Virtex 5 LX 50. La fréquence du circuit est de 188.83 MHz. La recherche exhaustive est réalisée en 3 heures et 32 minutes.

2.4 Comparaison de la règle de l'automate cellulaire trouvée avec des générateurs souvent utilisés en VLSI

Chaque CA est initialisé par le même germe, le vecteur 64 bits

$$(28FBB38AC1D8114A)_{16}.$$

La règle optimale trouvée est

$$(AAD32197)_{16} = (10101010110100110010000110010111)_2. \quad (4\text{-}3)$$

Celle-ci a été trouvée pour un seul vecteur d'initialisation donné. Ce travail n'a pas pour objectif de donner la règle optimale d'un CA pour tous les vecteurs d'initialisation possibles. L'idée est d'étudier le gain apporté par l'utilisation d'un CA ayant un nombre de voisins supérieur à quatre.

2.4-1 Entropie

La figure 4-12 compare l'entropie 8 bits du CA optimal à 5 voisins trouvé, d'un CA30 [Wol86] à 64 cellules,d'un LFSR de longueur 64 (LFSR64) [GA07] et du CA1D à 4 entrées de longueur 64 (règle $CA639_{16}$) [STCS02]. Les lignes représentées par des signes +, ∘ et □ correspondent respectivement au calcul de l'entropie pour une séquence de 1×10^4, 2×10^4 et 2×10^5 bits aléatoires. Le CA à 5 entrées converge plus rapidement vers l'entropie maximale.

Le tableau 4-2 résume les valeurs d'entropie obtenues lorsqu'une séquence de 2×10^5 bits est analysée et montre que le CA à 5 entrées possède l'entropie la plus élevée, 7.9942 bits. De plus, celle-ci est très proche de l'entropie maximale $H_{max} = 8$.

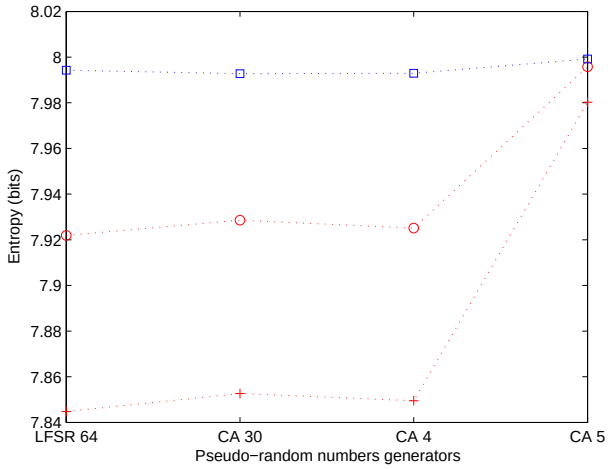

Figure 4-12 – La figure compare l'entropie 8 bits du CA optimal à 5 voisins trouvé, d'un CA30 [Wol86] à 64 cellules,d'un LFSR de longueur 64 (LFSR64) [GA07] et du CA1D à 4 entrées de longueur 64 (règle $CA639_{16}$) [STCS02]. Les lignes représentées par des signes +, ∘ et □ correspondent respectivement au calcul de l'entropie pour une séquence de 1×10^4, 2×10^4 et 2×10^5 bits aléatoires. Le CA à 5 entrées converge plus rapidement vers l'entropie maximale.

PRNG	Entropy (bits)
LFSR64	7.9942
CA 30	7.9927
CA 639_{16}	7.9929
CA $AAD32197_{16}$	7.9991

Tableau 4-2 – Le tableau résume les valeurs d'entropie obtenues lorsqu'une séquence de 2×10^5 bits est analysée et montre que le CA à 5 entrées possède l'entropie la plus élevée, 7.9942 bits. De plus, celle-ci est très proche de l'entropie maximale $H_{max} = 8$.

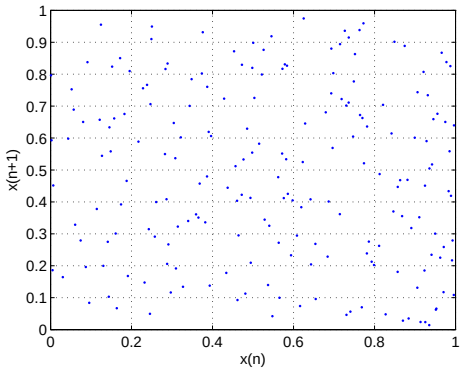

Figure 4-13 – Répartition en deux dimensions d'une séquence aléatoire de 400 entiers 32 bits générés par le CA à 5 voisins.

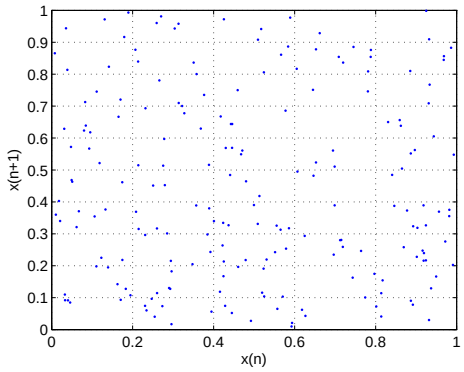

Figure 4-14 – Répartition en deux dimensions d'une séquence aléatoire de 400 entiers 32 bits générés par le CA à 4 voisins.

2.4-2 Distribution des bits aléatoires

Conclure l'évaluation par l'analyse seule de l'entropie serait une erreur. Il est nécessaire de tester la corrélation de la séquence aléatoire produite par les PRNG. Les figures 4-13, 4-14, 4-15 et 4-16 présentent graphiquement la répartition des bits aléatoires lorsque les PRNG génèrent une séquence aléatoire de 400 entiers 32 bits. Les points sont formés à partir de deux bits aléatoires consécutifs. Comme l'indiquent les figures, la répartition des points est uniforme. La même conclusion est faite en visualisant les points en trois dimensions (voir les figures 4-17, 4-18, 4-19 et 4-20).

La même expérimentation est réalisée lors de l'analyse d'une séquence aléatoire plus importante, 2000 entiers 32 bits. Les figures 4-21, 4-22, 4-23 et 4-24 présentent la répartition de bits aléatoires en deux dimensions. La figure 4-21 montre que la répartition des bits aléatoires produits par le CA à 5 voisins n'est pas valable en comparaison des autres PRNG. Ceci est validé par l'analyse des bits aléatoires en trois dimensions, voir les figures 4-25, 4-26, 4-27 et 4-28.

Ceci indique un problème dans la période du générateur et au niveau de la corrélation

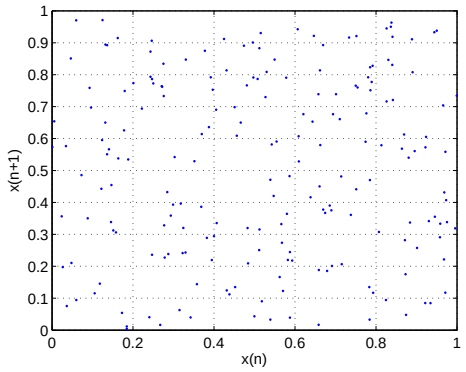

Figure 4-15 – Répartition en deux dimensions d'une séquence aléatoire de 400 entiers 32 bits générés par le CA30.

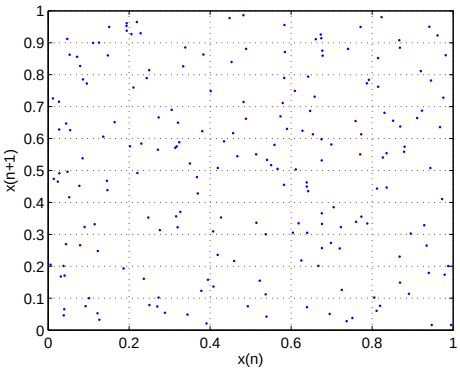

Figure 4-16 – Répartition en deux dimensions d'une séquence aléatoire de 400 entiers 32 bits générés par le LFSR64.

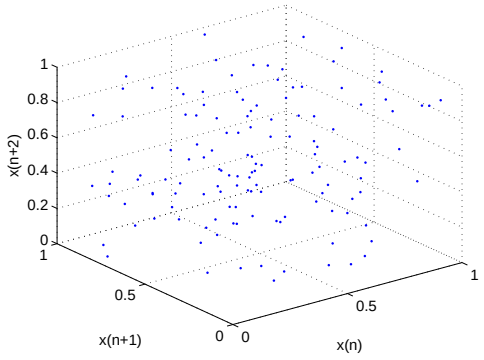

Figure 4-17 – Répartition en trois dimensions d'une séquence aléatoire de 400 entiers 32 bits générés par le CA à 5 voisins.

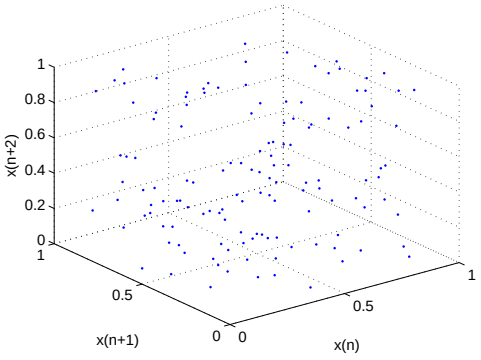

Figure 4-18 – Répartition en trois dimensions d'une séquence aléatoire de 400 entiers 32 bits générés par le CA à 4 voisins.

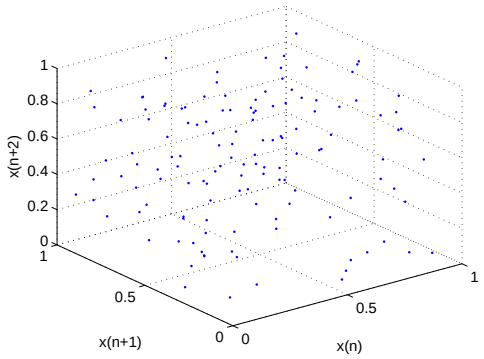

Figure 4-19 – Répartition en trois dimensions d'une séquence aléatoire de 400 entiers 32 bits générés par le CA30.

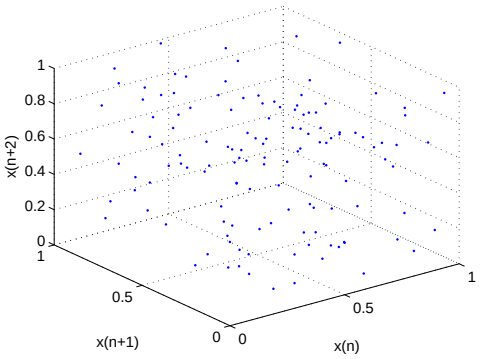

Figure 4-20 – Répartition en trois dimensions d'une séquence aléatoire de 400 entiers 32 bits générés par le LFSR64.

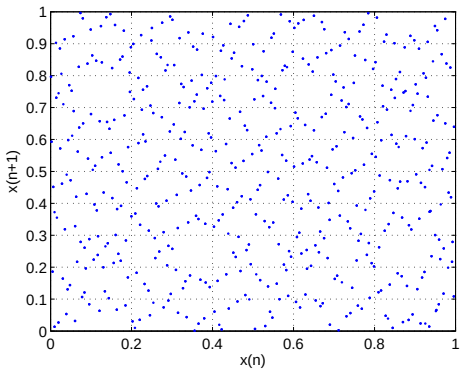

Figure 4-21 – Répartition en deux dimensions d'une séquence aléatoire de 2×10^4 entiers 32 bits générés par le CA à 5 voisins.

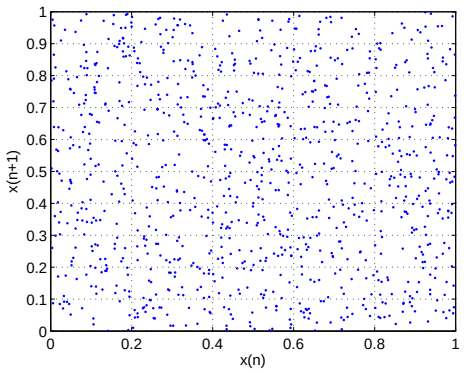

Figure 4-22 – Répartition en deux dimensions d'une séquence aléatoire de 2×10^4 entiers 32 bits générés par le CA à 4 voisins.

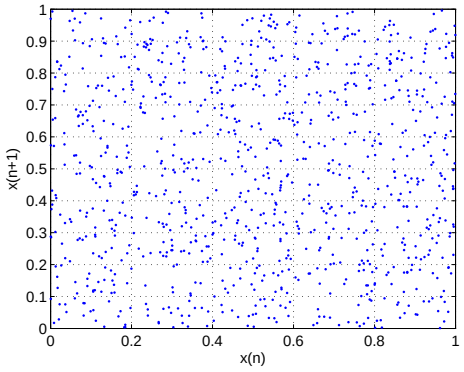

Figure 4-23 – Répartition en deux dimensions d'une séquence aléatoire de 2×10^4 entiers 32 bits générés par le CA30.

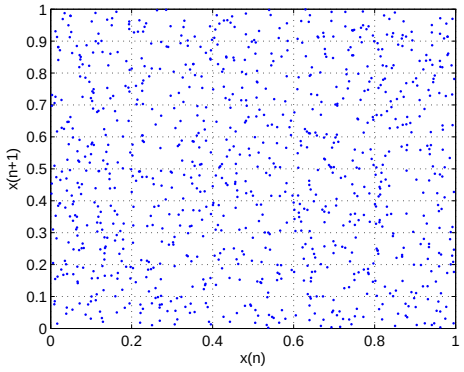

Figure 4-24 – Répartition en deux dimensions d'une séquence aléatoire de 2×10^4 entiers 32 bits générés par le LFSR64.

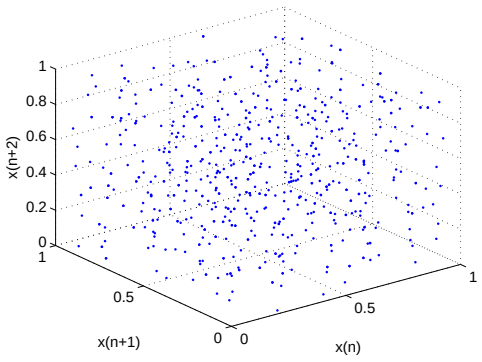

Figure 4-25 – Répartition en trois dimensions d'une séquence aléatoire de 2×10^4 entiers 32 bits générés par le CA à 5 voisins.

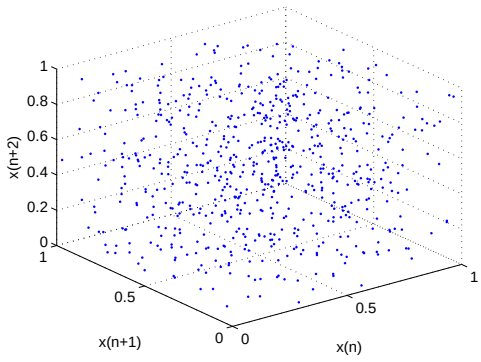

Figure 4-26 – Répartition en trois dimensions d'une séquence aléatoire de 2×10^4 entiers 32 bits générés par le CA à 4 voisins.

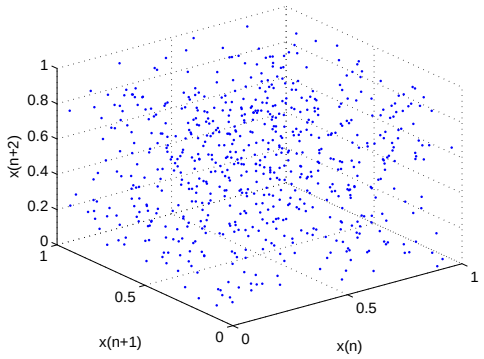

Figure 4-27 – Répartition en trois dimensions d'une séquence aléatoire de 2×10^4 entiers 32 bits générés par le CA30.

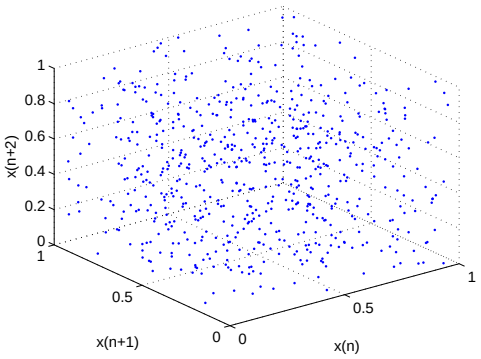

Figure 4-28 – Répartition en trois dimensions d'une séquence aléatoire de 2×10^4 entiers 32 bits générés par le LFSR64.

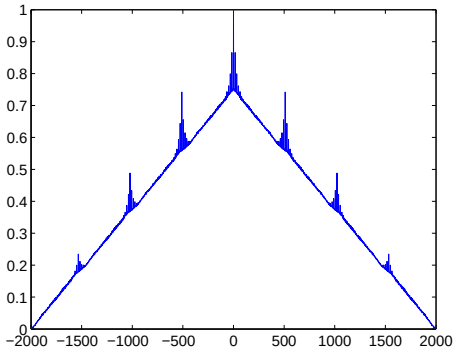

Figure 4-29 – Estimation de la fonction d'autocorrélation du CA à 5 voisins.

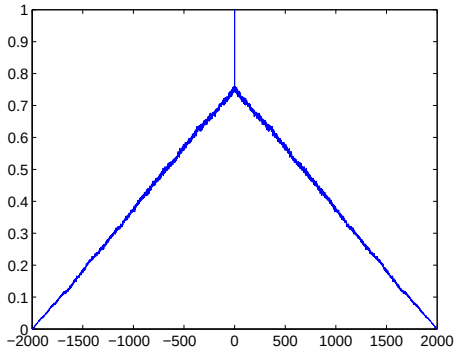

Figure 4-30 – Estimation de la fonction d'autocorrélation du CA à 4 voisins.

de la séquence binaire produite. Les fonctions d'autocorrélation des séquences aléatoires sont estimées par

$$R_{xx}(m) = \frac{1}{N - |m|} \sum_{n=0}^{N-m-1} x_{n+m} x_n. \tag{4-4}$$

Les résultats de l'estimation pour $m \in \{-2000, \ldots, 2000\}$ sont calculés pour les quatre PRNG et présentés dans les figures 4-29, 4-30, 4-31 et 4-32. La figure 4-29 montre que la séquence produite par le CA à 5 voisins est périodique, de période environ égale à 500 entiers 32 bits. La période des autres PRNG n'est pas visualisable à partir des figures 4-30, 4-31 et 4-32. Cependant, nous pouvons conclure que les fonctions d'autocorrélation des trois PRNG correspondent à la forme attendue.

2.4-3 Conclusion

Le CA à 5 voisins possède un débit de 429.15 Mb/s au sein d'une cible FPGA Virtex 5 LX50. Si le débit du générateur est élevé, les propriétés statistiques ne sont pas suffisantes

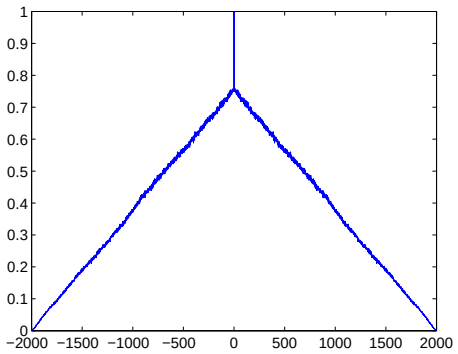

Figure 4-31 – Estimation de la fonction d'autocorrélation du CA30.

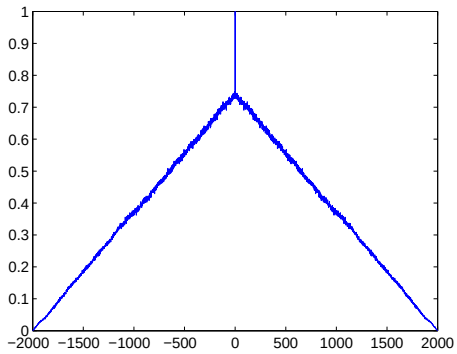

Figure 4-32 – Estimation de la fonction d'autocorrélation du LFSR64.

pour le qualifier de PRNG. La méthode utilisée est donc critiquable. La batterie FIPS 140-1 n'est pas suffisamment drastique pour tester la qualité aléatoire d'un PRNG. La périodicité du générateur est faible et la séquence est fortement corrélée par rapport aux autres générateurs testés. Une alternative serait d'utiliser des tests statistiques très efficaces en terme d'évaluation d'une séquence aléatoire. La prochaine section de ce chapitre étudie le coût matériel qu'engendrerait une telle implantation.

Une seconde conclusion consiste à dire que baser l'analyse sur le calcul de l'entropie 8 bits est insuffisant. Il est nécessaire d'augmenter la longueur en bits des blocs analysés (16 bits ou plus). Dans cette section le calcul de l'entropie est implanté de manière très simple, par l'utilisation de mémoire. L'augmentation de la taille imposera d'utiliser une méthode plus élégante pour effectuer le calcul de l'entropie.

3 Evaluation du coût matériel engendré par l'implantation de tests statistiques très performants

Les batteries Diehard, TestU01 et NIST sont les références du domaine lors de la validation d'un PRNG. Celles-ci sont constituées d'un grand nombre de tests statistiques, coûteux en terme de mémoire et de ressources arithmétiques. Par conséquent, implanter celles-ci en matériel est peu envisageable.

Cependant, après avoir étudié la corrélation des tests statistiques Diehard, [MT02] présente une version condensée de la batterie complète. La version comporte trois tests statistiques : le *GCD test*, le *gorilla test* et le *birthday spacings test*. A notre connaissance, aucune autre publication n'a testé la corrélation entre les tests statistiques d'une batterie. [MT02] a sélectionné les tests statistiques les plus difficiles à satisfaire par un générateur parmi ceux utilisés dans Diehard. D'après [MT02], si un PRNG réussit les trois tests statistiques, il passera certainement la batterie complète. L'objectif de cette partie est d'étudier le coût matériel engendré par l'implantation des trois tests. Ceci permettra d'envisager

l'utilisation des tests afin de trouver le CA à cinq voisins optimal.

3.1 *Birthday test*

3.1-1 Présentation

Le principe du test est de tirer aléatoirement m dates d'anniversaires dans une année comportant n jours. Les jours d'anniversaires sont classés par ordre croissant. Ensuite, les écarts entre les jours d'anniversaires sont calculés. Le nombre d'écarts revenant plus d'une fois (j) est comptabilisé. La variable aléatoire j suit une loi de Poisson de moyenne $\lambda = m^3/(4n)$. La version la plus stricte du test est obtenue pour $n = 2^{32}$, $m = 2^{12} = 4096$ et $\lambda = 4$.

La variable aléatoire j appartient à l'ensemble $\{0, 1, 2, 3, ..., 9, \geq 10\}$. Au delà, j est considérée être égale à 10. Un échantillon de 5×10^3 valeurs de j est relevé. Le nombre d'occurrences des j possibles est comptabilisé. Un test du χ^2 est ensuite effectué sur les occurrences de j

$$\chi^2_{10} = \sum_{i=0}^{10} \frac{(\#_i - \text{attendue}_i)^2}{\text{attendue}_i} \tag{4-5}$$

où $\#_i$ et attendue_i désignent respectivement le nombre d'occurrences rencontrées et attendues par la variable aléatoire $j = i$.

Deux hypothèses sont posées :
- H_0 : l'échantillon des 5×10^3 valeurs de j suit une loi de Poisson ;
- H_1 : l'échantillon des 5×10^3 valeurs de j ne suit pas une loi de Poisson.

Pour une probabilité de fausse alarme $\alpha = 0.05$, χ^2_{10} doit appartenir à l'intervalle de confiance $[3.247, 20.483]$ pour satifaire l'hypothèse H_0.

Exemple de déroulement

Un exemple de fonctionnement est donné ci-dessous. Dans cet exemple, $m = 10$ et $n = 2^{32}$.
- Le tableau des dates d'anniversaire est donné ci-dessous. La première cellule du tableau correspond à la cellule d'indice 0.

256	54	3	128	3	14	128	52	256	13

- Le tableau est ensuite trié.

3	3	13	14	52	54	128	128	256	256

- Les écarts entre les éléments consécutifs du tableau sont calculés. L'écart entre deux cellules est obtenu en soustrayant la valeur de la cellule de rang i à la cellule de rang $i-1$. Le calcul est effectué à partir de la dernière cellule jusqu'à la cellule de rang 1.

3	0	10	1	38	2	74	0	128	0

– Le tableau est de nouveau trié suivant les écarts croissants.

0	0	0	1	2	3	10	38	74	128

– Puis, le nombre d'écarts similaires est déduit à partir du tableau précédent. Le résultat est noté nb, ici $nb = 2$.

3.1-2 Implantation matérielle

Une première unité de traitement reçoit les bits aléatoires et regroupe ceux-ci en entiers 32 bits. Dans un premier temps, 4096 entiers sont enregistrés et rangés en série par ordre croissant. Ensuite, les entiers sont reclassés par ordre croissant. Les écarts entre les entiers successifs sont calculés puis triés. Une fois les 4095 écarts calculés, les écarts sont reclassés suivant l'ordre croissant. Le nombre d'écarts successifs égaux est comptabilisé. Le résultat est transmis en sortie de l'unité de traitement.

Deux blocs de tri sont ici nécessaires. Le premier réalise le tri des entiers 32 bits reçus. Le second bloc range les écarts entre les entiers préalablement rangés par ordre croissant. Quatre autres blocs constituent cette unité. Un bloc gère la réception en série des bits aléatoires et leur mise en vecteur de 32 bits. Un autre bloc calcule la différence entre les entiers triés afin de trouver les écarts. Un bloc teste l'égalité des écarts successifs. Le dernier bloc permet de séquencer l'unité de traitement.

Bloc de tri

Le bloc essentiel lors de la conception matérielle du test statistique est le tri d'une mémoire (ou tableau). Par convention, le tri est réalisé dans l'ordre croissant. Trier un tableau est une opération très coûteuse en terme de complexité, de temps et de surface. Dans la littérature, le tri a fait l'objet de plusieurs études [Knu73]. Lorsque celui-ci est réalisé à partir d'une mémoire remplie, un algorithme utilisant les papillons de la FFT rapide ou un tri rapide (*quicksort*) représente une bonne solution. Ici, le contexte d'utilisation du test statistique est différent. Tous les 32 cycles, un nouveau mot de 32 bits est disponible. Il est dommage de remplir une mémoire pour ensuite effectuer le tri. [LDH03], [HL00] et [BC98] présentent des méthodes de tri série.

La méthode de tri choisie est celle décrite dans [BC98]. Comme le montre la figure 4-33, celle-ci consiste à placer plusieurs cellules en série. Chaque cellule est constituée de deux registres 32 bits permettant de stocker deux dates d'anniversaires. Un multiplexeur est disposé entre deux cellules voisines et réalise le tri. Tous les 32 cycles, le nouvel entier reçu est comparé au contenu de la première cellule. Au cycle d'horloge suivant, l'entier le plus grand de chaque cellule est transmis au banc de registre suivant. Lors du reclassement de la mémoire, le même principe de fonctionnement est utilisé dans le sens inverse. À chaque

FPGA ciblé	Xilinx Virtex 5 XC5VSX50T-3
Nombre total de LUT	47106/32640 (144%)
Nombre BRAM	69/132 (52%)
Fréquence Maximale (MHz)	162.5 MHz

Tableau 4-3 – Résultat de l'implantation du bloc de tri dans un FPGA Xilinx Virtex 5 SX50.

cycle, l'entier inférieur de la cellule i est transmis à la cellule $i-1$.

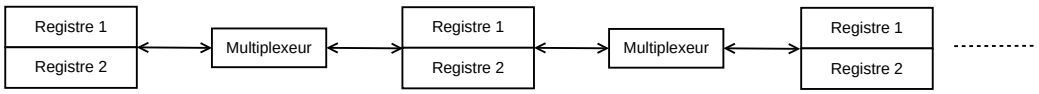

Figure 4-33 – Architecture pour le tri en série selon [BC98].

En pratique, il est préférable d'augmenter la taille des bancs de registres. Ainsi, ceux-ci sont facilement implantés au sein de blocs RAM. Des mémoires double ports sont utilisées (deux lectures ou une écriture par cycle possibles). Le problème de placement routage est ainsi diminué. Le tableau 4-3 résume le résultat de synthèse du bloc de tri au sein d'un FPGA Xilinx Virtex 5 SX50. Le coût en nombre de LUT et de bloc RAM est très important et nécessiterait d'utiliser une mémoire RAM externe. Le calcul du χ^2 est tout de même détaillé dans le prochain paragraphe.

3.1-3 Calcul du test du χ^2

Lors de l'implantation du *birthday test* au sein d'un FPGA, l'équation (4-5) doit être calculée en virgule fixe. Celle-ci est une somme de 11 valeurs intermédiaires. La valeur maximale contenue par une classe est égale à 5×10^3. Étant donné que cette grandeur est un entier positif, celui-ci est codé sur 13 bits. Il a été vu précédemment que le résultat doit appartenir à l'intervalle de confiance $[3.247, 20.483]$ pour une probabilité de fausse alarme de 5%. Il est possible de calculer les valeurs maximales et minimales admissibles pour chacune des 11 classes de manière similaire à la procédure utilisée pour le *poker test* du FIPS 140-2, voir partie 2.2-2. La relation à satisfaire est donnée par l'inéquation

$$3.247 \leq \sum_{i=1}^{k} \frac{(\#_i - \text{attendue}_i)^2}{\text{attendue}_i} \leq 20.483. \tag{4-6}$$

La première étape est de vérifier la valeur contenue par chaque classe. Si celle-ci n'appartient pas au domaine requis, le calcul est arrêté, le test statistique est un échec.

Afin d'éviter le calcul dans l'équation 4-6 de la division, cette opération est remplacée par une multiplication par les valeurs inverses des coefficients attendue$_i$. Par conséquent, pour chaque classe, les coefficients $-$attendue$_i$ (pour réaliser une addition) et leurs valeurs inverses doivent être mémorisés. Le graphe flot de signal est présenté dans la figure 4-34.

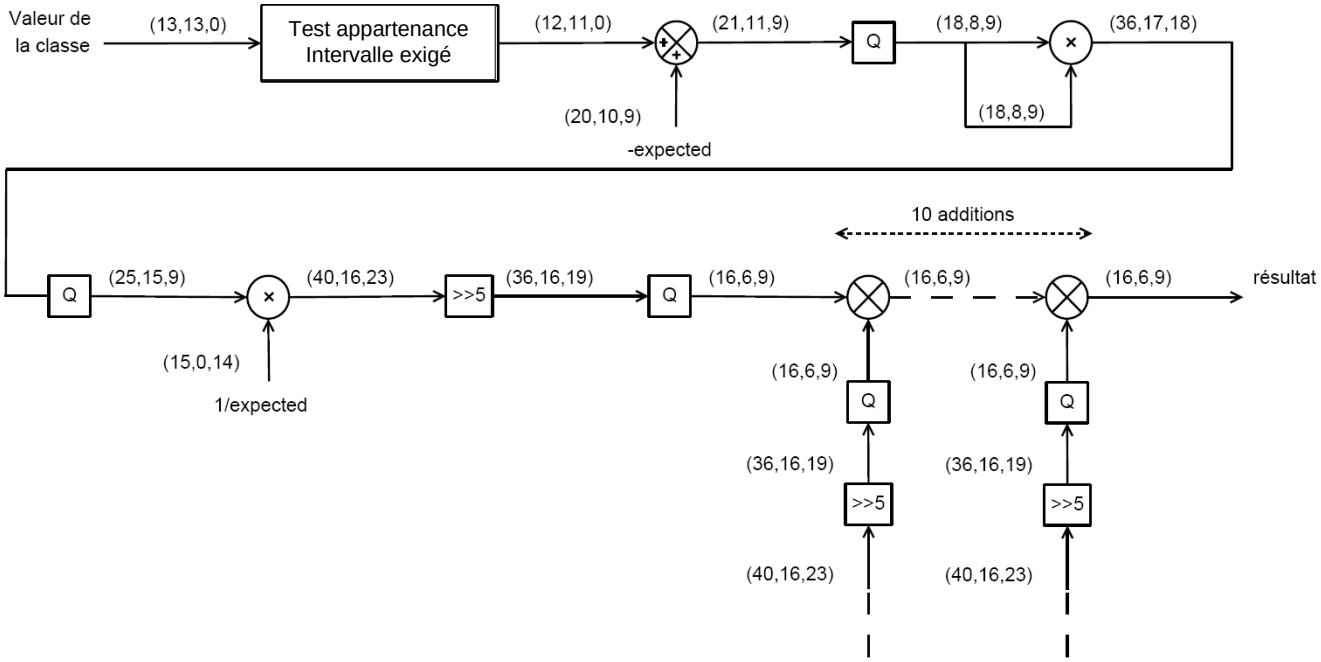

Figure 4-34 – Graphe flot de signal du test du χ^2 dans le *birthday test*.

3.2 *Gorilla test*

3.2-1 Présentation

Le *gorilla test* permet de tester la périodicité d'un générateur de nombres aléatoires. Celui-ci est basé sur l'utilisation d'un fichier disposant de $2^{26} + 25 = 67108889$ entiers 32 bits. Sur chaque entier, un bit spécifique est prélevé, par exemple le bit i. Une séquence binaire constituée du $i^{\text{ième}}$ bit de chaque entier 32 bits est créée. La séquence binaire résultante comporte $2^{26} + 25 = 67108889$ bits aléatoires.

Soit w, le nombre de mots de 26 bits n'apparaissant pas dans la séquence binaire. w suit approximativement une loi Normale de moyenne 24687971 et de variance 4170. En se ramenant à la loi Normale $N(0, 1)$, la probabilité p_{value} est déterminée. Ce test est réalisé pour i variant de 0 à 31 (chaque position binaire de l'entier aléatoire 32 bits). Au total, trente deux valeurs de probabilité sont obtenues. Le test d'*Anderson-Darling-Kolmogorov-Smirnov* permet de déterminer une seule probabilité p_{value} à partir des 32 valeurs de probabilité. Le test échoue si la valeur de probabilité p_{value} déterminée est proche de 0 ou de 1.

Pour une position binaire, le test nécessite d'utiliser une mémoire de type CAM de taille 2^{32} bits. Ceci est difficilement envisageable sans avoir recours à l'utilisation d'une mémoire externe.

Par conséquent, l'objectif est d'implanter un test proche du fonctionnement du *gorilla test* mais nécessitant moins de ressources mémoires. Le *gorilla test* est en réalité une version stricte de l'*overlapping* 20-*tuples bitstream test* [MT02]. Dans la littérature, l'*overlapping* 20-*tuples bitstream test* et le *monkey test* désignent le même test statistique [MZ93]. D'après [MT02], le *monkey test* est le test le plus difficile à réussir pour beaucoup de générateurs.

3.2-2 *Overlapping 20-tuples bitstream test*

L'*overlapping 20-tuples bitstream test* (OMTBT) travaille sur une séquence binaire composée de 1310721 entiers 32 bits. L'alphabet du test est $\{0, 1\}$. L'OMTBT consiste à générer des mots de 20 lettres avec recouvrement. Au total, 2^{16} séquences binaires de 20 bits sont analysées. Le nombre de mots manquants est comptabilisé. Le résultat suit une loi Normale $\mathcal{N}(141909.33, 428)$. En se ramenant à une loi Normale réduite $\mathcal{N}(0, 1)$, une probabilité p_{value} est calculée.

3.2-3 Implantation matérielle

Une mémoire accessible par son contenu de taille $2^{20} \times 1$ bits est ici nécessaire. Celle-ci est réalisée à partir d'une mémoire de taille $2^{15} \times 32$ bits. Une fois le nombre de mots manquants obtenu, l'équation

$$z = \frac{\text{Nombre mots manquants } - 141909.33}{428} \tag{4-7}$$

est calculée. Pour une probabilité de fausse alarme $\alpha = 5\%$, le résultat doit appartenir à l'intervalle de confiance $[-1.96; 1.96]$. Ainsi, la variable aléatoire *Nombre mots manquants* doit appartenir à $[141070.45; 142748.21]$.

3.3 *GCD test*

3.3-1 Présentation

Le générateur de nombres aléatoires fournit 20^7 entiers 32 bits. Le test détermine :

- le plus grand diviseur commun entre deux nombres consécutifs, le pgcd. D'après [MT02], soit X la variable aléatoire correspondant au résultat du pgcd, la probabilité de X est définie de la manière suivante :

$$Pr[X = i] = \frac{c}{i^2}, \tag{4-8}$$

où

$$c = \frac{6}{\pi^2}. \tag{4-9}$$

Un test du χ^2 compare les valeurs de pgcd relevées (10^7 valeurs au total) à celles attendues. Le résultat suit un χ^2 à 99 degrés de liberté. Le test renvoie une valeur p_{value}.

- le nombre d'itérations nécessaire à l'algorithme d'Euclide pour déterminer le calcul du pgcd : kp. Trente-trois valeurs de kp sont possibles : $2 < kp < 36$. kp suit approximativement une loi Binomiale de paramètres $n = 50$ et $p = 0.376$. A partir des relevés de kp (10^7 valeurs), un test d'adéquation est réalisé. Le test compare les valeurs kp observées avec les valeurs attendues. Le résultat obtenu suit un test du χ^2 à 32 degrés de liberté. Le test détermine une valeur de probabilité p_{value}.

Le test échoue si une des valeurs de probabilité p_{value} calculées est inférieure à 0.025 ou supérieure à 0.975.

3.3-2 Implantation matérielle

L'implantation matérielle du *GCD test* nécessite de se concentrer sur l'algorithme d'Euclide. La méthode utilisée pour calculer le test du χ^2 sur les valeurs kp et pgcd est similaire à celle étudiée dans la partie 3.1-3 et n'est pas présentée ici.

Algorithme d'Euclide

L'algorithme d'Euclide permet de déterminer le plus grand diviseur commun (PGCD) de deux entiers a et b. L'algorithme est basé sur l'équation

$$a = b.q + r_1, r_1 < b. \tag{4-10}$$

Ainsi, PGCD(a,b)=PGCD(b,r). Ceci est vrai car l'ensemble des diviseurs communs à a et à b est aussi l'ensemble des diviseurs communs à b et à r. En réalisant la division euclidienne de a par b, puis de b par r_1, le nouveau reste r_2 est obtenu avec $r_1 > r_2$. Puis la division euclidienne de r_1 par r_2 est effectuée obtenant ainsi le nouveau reste r_3. En réitérant le procédé, une suite décroissante (r_n) est engendrée. Le processus s'arrête lorsque le nouveau reste calculé à partir de la division euclidienne est nul. L'algorithme est détaillé dans [PFTV88]. Lors de l'implantation de l'algorithme d'Euclide, la division euclidienne est réalisée par un calcul de modulo. Réaliser un opérateur modulo en matériel est difficile pour certaines valeurs.

Le *GCD test* utilise le nombre d'itérations dans l'algorithme d'Euclide. Une solution permettant d'obtenir le même nombre d'itérations nécessaire au calcul du PGCD qu'avec l'algorithme d'Euclide doit être trouvée. La solution est de réaliser des divisions successives afin d'obtenir le reste de la division à chaque étape. Plusieurs ouvrages présentent l'implantation de diviseurs en virgule fixe : [Par00], [OF01], [EL04] et [Zim]. Ici, le choix a été d'utiliser l'IP *Divider Generator* [Xil09] fournie par Xilinx. Un diviseur pipelinable permettant de lancer une nouvelle division par cycle et renvoyant un résultat tous les 34 cycles est obtenu.

FPGA ciblé	Xilinx Virtex 5 XC5VSX50T-3
Nombre total de LUT	6286/32640 (19%)
Nombre BRAM	6/132 (4.5%)
Fréquence Maximale (MHz)	107 MHz

Tableau 4-4 – Résultat de l'implantation du *gcd test* dans un FPGA Xilinx Virtex 5 SX50.

Afin de calculer le nombre de diviseurs requis, il est nécessaire de connaître le nombre maximal de divisions successives pour calculer le PGCD d'un couple (a, b) par l'algorithme d'Euclide. D'après [Knu97], lorsque $a < b < N$, ce nombre est égal à

$$nbDiv = \lfloor \log_{\phi} (3 - \phi) N \rfloor \approx \lfloor 2.078 \ln N + 0.6723 \rfloor. \tag{4-11}$$

Ainsi, pour $N = 2^{32}$, le nombre maximal de divisions successives lors d'un calcul de PGCD est égal à 46.

Un nouveau calcul de PGCD est réalisé tous les 64 cycles d'horloges. Une division est lancée tous les 35 cycles d'horloges (34 cycles de latence pour le calcul d'une division et un cycle pour évaluer le résultat de celle-ci). Le nombre de calcul de PGCD à exécuter pendant les 46 divisions successives (dans le pire cas) est égal à

$$\left\lceil \frac{46 \times 35}{64} \right\rceil = 26. \tag{4-12}$$

Ainsi, une seule division doit être lancée à chaque cycle d'horloge. Ceci permet d'utiliser un seul diviseur. Cependant, étant donné que 26 calculs de PGCD sont réalisés en parallèle à partir du seul diviseur, une mémoire de taille 26×32 bits est nécessaire pour sauvegarder les valeurs de diviseur.

Performances

Le circuit est synthétisé et placé-routé au sein d'un FPGA Xilinx Virtex 5 SX50T. Le tableau 4-4 résume les performances du circuit. La fréquence maximale du circuit est de 107 MHz et occupe environ 19% des LUT et 6 BRAM.

4 Conclusions

Un grand nombre d'applications nécessitent d'utiliser un générateur à très haut débit et peu coûteu en termes de ressources matérielles. L'objectif de ce chapitre était de rechercher un générateur pseudo-aléatoire profitant de la structure parallèle d'un FPGA. Les automates cellulaires possèdent une structure intéressante pour atteindre des débits maximum. Cependant, malgré leur simplicité, prédire le comportement d'un CA en fonction de la règle utilisée est très difficile. Une solution pour trouver la règle optimale est d'utiliser

une recherche exhaustive. Cependant, le nombre d'entrées d'une cellule augmentant, une recherche exhaustive via des tests logiciels est difficilement envisageable.

Après description du principe de fonctionnement d'un automate cellulaire, une méthode de recherche exhaustive est proposée afin de trouver la règle optimale d'un CA à 5 voisins. Celle-ci est basée sur l'implantation des tests statistiques FIPS 140-1. Le faible coût matériel des tests permet l'analyse d'un grand nombre de règles en parallèle. De plus, le temps de recherche est optimisé en générant, directement en matériel, l'ensemble des combinaisons possibles. Les premiers résultats sont intéressants.

Cependant, à partir de l'analyse de la corrélation et de la périodicité des séquences binaires produites par le générateur, des défaillances non-négligeables sont détectées. Le générateur ne peut pas être qualifié de PRNG. Deux conclusions sont possibles, soit il n'existe aucune règle optimale de CA à cinq voisins, soit la méthode d'évaluation n'est pas suffisamment drastique. Nous optons pour cette seconde hypothèse. La solution proposée est d'utiliser des tests statistiques très performants.

La dernière sous-section du chapitre présente le coût matériel engendré par l'implantation des tests les plus performants de la batterie Diehard [MT02]. Les tests nécessitent l'utilisation d'un grand nombre de ressources mémoires, limitant leur implantation en interne d'un FPGA. Une solution envisageable serait d'utiliser des mémoires externes.

Conclusions et perspectives

Synthèse des travaux

Les travaux présentés dans ce document sont dédiés à la définition et au développement de générateurs aléatoires uniformes et gaussiens à très haut débit. Les nombres aléatoires sont obtenus à partir d'un générateur hybride composé d'un TRNG et d'un PRNG. Sélectionner un TRNG est délicat car un grand nombre de générateurs existent dans la littérature. De plus, les générateurs développés sont rarement comparés aux TRNG déjà existants. Enfin, la forte dépendance d'un TRNG à la qualité de l'implantation, à l'environnement extérieur et au circuit ciblé rend l'analyse des performances des TRNG difficile.

Tout d'abord, après avoir sélectionné trois TRNG pour leurs performances a priori et la possibilité d'implanter ceux-ci au sein d'une multitude de circuits reconfigurables, les propriétés statistiques des sorties des générateurs ont été évaluées. L'étude a été réalisée à partir de l'implantation des tests statistiques du FIPS 140-2. Les performances du composant permettent d'analyser la qualité des TRNG en temps réel, directement sur cible FPGA. Les TRNG implantés au sein du même circuit reconfigurable sont évalués en fonction de la température exercée sur le FPGA ainsi que de l'activité parasite appliquée autour de la source d'aléa. Les expérimentations montrent que le comportement des générateurs est peu sensible aux variations de températures appliquées sur le circuit. Au contraire, l'activité parasite entraîne une dégradation non négligeable des performances du TRNG proposé par Dichtl et al. [DG07]. Le générateur présenté par Sunar et al. [SMS07] semble être le plus robuste des trois TRNG testés. Le générateur de Vasyltsov et al. [VHKK08] échoue fortement aux tests statistiques. Ceci provient du fait que le signal métastable possède un biais non négligeable car, tandis qu'en technologie ASIC il est possible de faire tendre le signal vers $V_{dd}/2$, la métastabilité est plus difficile à obtenir dans un FPGA et un biais existe.

L'expérimentation a montré la variation de la qualité d'un TRNG en fonction de l'activité parasite autour du générateur. Malgré la robustesse du TRNG proposé par Sunar et al. [SMS07], certaines séquences aléatoires sont parfois qualifiées de défaillantes par les tests du FIPS 140-2. Cependant, les tests statistiques du FIPS 140-2 ne font aucune différence entre l'évaluation d'une séquence issue de l'échantillonnage du bruit ou de la sortie après post-traitement. Malgré les bons résultats obtenus par le TRNG de Sunar et al. [SMS07], la complexité de la fonction résiliente (code BCH) peut cacher certaines défaillances du générateur.

L'évaluation par les tests du FIPS 140-2 a permis de retenir les générateurs de Dicht et al. et celui de Sunar et al. Les générateurs sont basés sur la gigue d'un ou plusieurs oscillateur(s) à anneau. Afin de montrer l'importance d'adapter le comportement d'un TRNG (i.e. débit) en fonction du circuit ciblé, de son activité et de l'environnement extérieur, des mesures réelles de la gigue cycle-à-cycle d'un oscillateur à anneau ont été réalisées. La variation de la température du FPGA a un impact négligeable sur les caractéristiques de la gigue cycle-à-cycle. Par contre, les mesures indiquent que la gigue varie en fonction du FPGA utilisé et de son activité. Lors d'une activité importante autour de l'oscillateur analysé, la partie déterministe est augmentée. Ceci laisse supposer que lorsque plusieurs RO sont routés de manière connexe, un couplage peut exister entre eux. Par conséquent, au sein d'un TRNG, l'activité des différents RO peut introduire de la corrélation entre les oscillateurs et réduire la qualité aléatoire du bruit échantillonné. De plus, les mesures montrent que l'oscillateur implanté au sein du FPGA Altera Stratix II est moins sensible à l'activité environnante qu'avec le FPGA Xilinx Virtex-4.

À partir de cette analyse, une méthodologie d'évaluation rapide et efficace est proposée. Celle-ci évalue à la fois les propriétés statistiques du signal DAS et de la séquence binaire en sortie du TRNG. Le débit maximal des générateurs est recherché en utilisant une PLL reconfigurable. La méthode basée sur l'utilisation d'une implantation efficace des tests statistiques proposés par l'AIS 31 et le FIPS 140-2 permet l'évaluation des générateurs en temps réel au sein d'un FPGA Xilinx Virtex 5 SX50. En sortie du signal DAS, les débits maximums des TRNG de Dichtl et al. et de Sunar et al. sont respectivement de 6.53 Mbits/s et de 8.59 Mbits/s. De même, en sortie des générateurs, les débits maximums sont respectivement de 5.5 Mbits/s et 6.53 Mbits/s. Le débit plus élevé des signaux DAS indique la sous optimalité des unités de post-traitement.
Dichtl et al. spécifiait un débit de 6.53 Mbits/s au sein d'un FPGA Altera Spartan-3. Ici, dans le FPGA Xilinx Virtex 5 ciblé, une diminution du débit du TRNG est obtenu. Sunar et al. précisait un débit de 2.5 Mbits/s sans un FPGA Xilinx Virtex 2. Le débit du TRNG est grandement amélioré au sein du FPGA ciblé (d'un facteur 2.61). Ceci montre l'intérêt d'utiliser la méthodologie d'évaluation pour caractériser le fonctionnement d'un générateur.

Si le débit du TRNG de Sunar et al. est plus élevé, l'évaluation de la surface des générateurs et de leur consommation montre que le TRNG est relativement coûteux. Par conséquent, le TRNG proposé par Dichtl et al. sera préféré au sein des circuits présentant

de fortes contraintes en surface et en consommation. Par ailleurs, la méthode proposée a permis d'optimiser les débits des TRNG au sein du FPGA visé.

L'évaluation de la qualité des TRNG a permis d'évaluer la performance des TRNG testés au sein du FPGA ciblé. De plus, l'expérimentation a montré que la qualité d'un générateur peut varier au cours du temps en fonction de la qualité d'implantation et de l'activité du circuit. Par conséquent, en vue de l'implantation d'un RNG hybride de qualité, des mécanismes de test en ligne doivent être mis en œuvre afin d'évaluer la qualité d'un TRNG. La suite du travail présente l'implantation d'un RNG hybride au sein d'un circuit reconfigurable. Le circuit a été développé en collaboration avec une entreprise spécialisée en cryptographie, dans le cadre du projet SPRING (*Shelf Proof Random Integrated Number Generator*). Le générateur est composé du TRNG de Sunar et al., d'un AES-128 agissant comme PRNG et de la batterie du FIPS 140-2. Celle-ci est utilisée afin de contrôler, en temps réel la qualité des séquences d'initialisation du PRNG. Cette analyse permet de détecter les défaillances éventuelles du TRNG.

Si les études précédentes ont permis de sélectionner un TRNG en vue d'une implantation FPGA, la réalisation d'un générateur vraiment aléatoire au sein d'un circuit dédié pose de nouvelles questions. Le travail présente le développement et la réalisation de deux RNG hybrides en technologie CMOS 130 nm.

Le premier circuit composé du TRNG de Schellekens et al. [SPV06], du FIPS 140-2 et d'un automate cellulaire à quatre voisins nous a permis de prendre en main la chaîne de développement ASIC. Les premiers résultats de validation montrent que le débit maximal du TRNG est de 1.25 Mbits/s. Cependant, l'évaluation de la gigue d'oscillateurs à anneau implantés sur le circuit montre que la gigue cycle à cycle dévie d'une loi Normale. La réception tardive du circuit ASIC au laboratoire n'a pas permis d'étudier en détail les performances du générateur. De plus amples analyses doivent être menées. Il sera nécessaire d'évaluer les propriétés statistiques du bruit échantillonné. Deux analyses seront effectuées. La première consistera à utiliser les tests statistiques conseillés par l'AIS 31 afin d'évaluer le signal DAS. La seconde analyse étudiera l'autocorrélation du bruit échantillonné à partir de la mémorisation d'un grand nombre de réalisations du signal. De plus, le circuit sera utilisé pour analyser le comportement du générateur proposé par Schellekens et al. [SPV06] en fonction d'attaques réalisées sur celui-ci (i.e. modulation de la tension d'alimentation).

Le second circuit est en cours de développement et sa fabrication est prévue pour Janvier 2010. Ce circuit a pour objectif d'étudier les performances de quatre TRNG implantés en ASIC. Le circuit est composé des TRNG présentés par Sunar et al. [SMS07], Vasyltsov et al. [VHKK08], Dichtl et al. [DG07] et Yoo et al. [YSKB]. Les tests statistiques proposés par l'AIS 31 et le FIPS 140-2 sont implantés. Le FIPS 140-2 permet de valider en temps réel la qualité de la séquence vraiment aléatoire produite par le TRNG sélectionné. La batterie de l'AIS 31 permettra d'étudier en détails les peformances des signaux DAS et des sorties des TRNG. La séquence analysée par le FIPS 140-2 est ensuite utilisée pour initialiser un générateur pseudo-aléatoire, un AES-128 fonctionnant en mode OFB. Les

nombres aléatoires uniformes sont générés à un débit de 2.57 Gbits/s.
De plus, le circuit génère des nombres aléatoires gaussiens et suivant une loi de Rayleigh. Les débits maximums en sortie des deux générateurs sont respectivement de 3.287 Gbits/s et 6.5375 Gbits/s.

La dernière partie du document présente une étude annexe réalisée au cours de la thèse consistant à rechercher un générateur pseudo-aléatoire optimisé selon un compromis qualité/coût matériel. L'objectif était de rechercher un générateur pseudo-aléatoire profitant de la structure "systolique" d'un FPGA. Les automates cellulaires (CA) possèdent une structure intéressante pour atteindre des débits maximums. Cependant, malgré leur simplicité, prédire le comportement d'un CA en fonction de la règle utilisée est très difficile. Une solution permettant la recherche exhaustive de la règle optimale d'un automate cellulaire à cinq voisins est présentée. Celle-ci est basée sur l'implantation des tests statistiques du FIPS 140-1. Le faible coût matériel des tests permet l'analyse d'un grand nombre de règles en parallèle. De plus, le temps de recherche est optimisé en générant, directement en matériel, l'ensemble des combinaisons possibles. Les premiers résultats sont intéressants. Cependant l'analyse de la corrélation et de la périodicité des séquences binaires produites par le générateur montre des défaillances non-négligeables. Le générateur ne peut pas être qualifié de PRNG. Deux conclusions sont possibles, soit il n'existe aucune règle optimale de CA à cinq voisins, soit la méthode d'évaluation n'est pas suffisamment drastique. Nous optons pour cette seconde hypothèse. La solution proposée est d'utiliser des tests statistiques très performants. Le coût matériel engendré par l'implantation des tests les plus performants de la batterie de Diehard [MT02] a été présenté. Les tests nécessitent l'utilisation d'un grand nombre de ressources mémoires, limitant leur implantation en interne d'un FPGA. Une solution envisageable serait d'utiliser des mémoires externes.

Perspectives

L'état de l'art présenté dans le chapitre I montre à quel point la génération de nombres vraiment aléatoires est une tâche primordiale au sein des RNG. Dans le cadre des travaux présentés dans ce document, plusieurs perspectives sont envisageables.

Parmi ces perspectives, nous pouvons citer l'implantation d'une mesure sur puce, temps réel de la source d'aléa. La plupart des TRNG sont basés sur la gigue d'un ou plusieurs oscillateurs. La méthode de contrôle présentée dans ce document utilise l'implantation efficace des tests du FIPS 140-2. Cependant, le FIPS 140-2 ne fait aucune différence entre l'analyse du signal DAS et de la séquence binaire en sortie de TRNG. De plus, malgré l'efficacité de l'implantation, celle-ci est relativement coûteuse par rapport à celle d'un TRNG. Enfin, pour maximiser l'efficacité de l'évaluation, un mécanisme de test au plus proche de la source d'aléa est nécessaire. Notre objectif est de proposer une mesure sur puce de la gigue, peu coûteuse en terme de surface et facilement implantable au sein d'un circuit reconfigurable. Plusieurs méthodes de mesure développées au sein de circuit ASIC existent dans la

littérature. La première technique consiste à sur-échantillonner le signal analysé, l'horloge en sortie de l'oscillateur à anneau. Cependant, afin d'obtenir une précision suffisante, par exemple 100 ps, une horloge très rapide est requise (10 GHz), limitant l'utilisation de la technique au sein des circuits VLSI. Une seconde alternative est d'utiliser un convertisseur temps/amplitude. Une mesure temporelle est obtenue à partir de la décharge d'une capacité. Cette technique présente l'inconvénient de nécessiter des composants analogiques tels qu'une source de courant, un interrupteur rapide et un convertisseur analogique numérique. Par conséquent, cette technique n'est pas réalisable au sein des circuits numériques actuels. La troisième méthode est de convertir une mesure temporelle en une grandeur numérique (*Time-to-Digital Converter* (TDC)). L'avantage des TDC est de pouvoir être totalement implantables au sein des circuits numériques. La *Vernier Delay Line* (VDL) utilise ces méthodes et permet de mesurer, en temps réel la fonction de répartition de la gigue. Plusieurs implantations au sein de circuit ASIC sont reportées dans la littérature, par exemple dans [HG02], [CR04], [SR99] et [SR04]. Cependant, en technologie FPGA, l'implantation d'une VDL est assez délicate car le contrôle des délais de routages est nécessaire à la précision de la mesure. La connaissance de la fonction de répartition de la gigue permettra d'évaluer si celle-ci correspond à celle attendue en théorie. Par ailleurs, il sera possible d'analyser l'importance des parties déterministes au sein du spectre de la gigue. Ceci permettra de détecter les possibles attaques réalisées à partir de modulation de l'alimentation.

L'utilisation d'une VDL est pertinente, cependant, en fonction de la latence de la mesure désirée, le circuit peut s'avérer coûteux en termes de surface. Une alternative afin de diminuer la surface du circuit est de mesurer l'écart type de la gigue. Les circuits proposés par [Hua06] et [FC09] sont en cours d'implantation.

Une fois validées, les mesures sur puce de la gigue seront implantées au sein du RNG hybride présenté dans ce document. La corrélation entre les résultats des tests statistiques et les mesures réalisées sera étudiée.

Une autre perspective est d'étudier à partir de l'utilisation des circuits OCHRE le comportement des TRNG implantés sur cible ASIC. Des attaques non-invasives sur l'alimentation du circuit sont prévues. Celles-ci ont été choisies en raison de leur impact sur la gigue des oscillateurs à anneaux (augmentation de la partie déterministe de la gigue).

Après ce travail, des optimisations du GRNG proposé seront réalisées. L'objectif sera d'automatiser la méthode d'évaluation permettant l'implantation des fonctions utilisées dans l'algorithme de Box-Muller. Les outils Sollya [1] et Gappa [2] seront utilisés pour automatiser la recherche des polynômes minimax et obtenir la précision en sortie du schéma de Horner de leur évaluation. Le code VHDL généré, les performances de l'implantation seront obtenues à partir de scripts de synthèse et placement-routage sous l'outil Xilinx ISE. À partir de contraintes de surface et/ou de débit et de précision, la configuration optimale sera sélectionnée.

1. http ://sollya.gforge.inria.fr/
2. http ://lipforge.ens-lyon.fr/www/gappa/

Ce mode d'évaluation permettra de proposer des améliorations du GRNG présenté et d'étudier le coût engendré par une augmentation de la valeur maximale atteinte en queue de gaussienne ($\max_\sigma$).

Enfin, la dernière perspective sera d'implanter en matériel des modèles de canaux de transmission plus complexes.

Bibliographie

[Abd03] Mustapha Abdulai. Inexpensive parallel random number generator for configurable hardware. Technical report, Arizona State University, Tempe Arizona, Summer Undergraduate Program in Engineering Research at Berkeley (SUPERB), 2003.

[ABS05] Alimohammad A., Cockburn B.F., and C. Schlegel. Area-efficient parallel white gaussian noise generator. In *Electrical and Computer Engineering, Canadian Conference on*, 2005.

[AFCS08] Amirhossein Alimohammad, Saeed Fouladi Fard, Bruce F. Cockburn, and Christian Schlegel. A compact and accurate gaussian variate generator. *IEEE Trans. Very Large Scale Integr. Syst.*, 16(5) :517–527, 2008.

[AIS01] AIS 31 : Functionality classes and evaluation methodology for physical random number generators. version 1 (25.09.2001). Technical report, 2001.

[And01] Ilachinski Andrew. *Cellular Automata : A Discrete Universe*. World Scientific Publishing Company, 2001.

[BB99] Vittorio Bagini and Marco Bucci. A design of reliable true random number generator for cryptographic applications. In *Proc. Workshop Cryptographic Hardware and Embedded Systems (CHES'99)*, pages 204–218, London, UK, 1999. Springer-Verlag.

[BC98] Marc Bumble and Lee Coraor. Architecture for a non-deterministic simulation machine. In *WSC '98 : Proceedings of the 30th conference on Winter simulation*, pages 1599–1606, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[BDG03] Emmanuel Boutillon, Jean-Luc Danger, and Adel Ghazel. Design of high speed awgn communication channel emulator. *Analog Integr. Circuits Signal Process.*, 34(2) :133–142, 2003.

[BDI86] J.C. Bic, D. Duponteil, and J.C. Imbeaux. *Eléments de Communications Numériques*. Dunod, 1986.

[BECH+06] H. Bar-El, Choukri, H., D. Naccache, M. Tunstall, and C. Whelan. The sorcerer's apprentice guide to fault attacks. *Proceedings of the IEEE*, 94(2) :370–382, 2006.

[BGL+03] Marco Bucci, Lucia Germani, Raimondo Luzzi, Alessandro Trifiletti, and Mario Varanonuovo. A high-speed oscillator-based truly random number source for cryptographic applications on a smart card ic. *IEEE Transactions on Computers*, 52(4) :403–409, 2003.

[BGL+06] M. Bucci, L. Giancane, R. Luzzi, M. Varanonuovo, and A. Trifiletti. A Novel Concept for Stateless Random Bit Generators in Cryptographic Applications. In *International Symposium on Circuits and Systems (ISCAS 2006), Island of Kos, Greece, May 21 - 24, 2006, Proceedings*, pages 317–320. IEEE Computer Society, May 2006. ISBN 0-7803-9390-2.

[Blu84] Manuel Blum. Independent unbiased coin flips from a correlated biased source : a finite state markov chain. In *FOCS*, pages 425–433, 1984.

[BM58] G. E. P. Box and M. E. Muller. A note on the generation of random normal deviates. *The Annals of Mathematical Statistics*, 29 :610–611, 1958.

[BMV06] Johannes Buchmann, Alexander May, and Ulrich Vollmer. Perspectives for cryptographic long-term security. *Commun. ACM*, 49(9) :50–55, 2006.

[BST03] B. Barak, R. Shaltiel, and E. Tomer. True random number generators secure in a changing environment. In *Proc. Workshop Cryptographic Hardware and Embedded Systems (CHES'03)*, 2003.

[CLLL06] Hong-Yu Chen, Wen-Jeng Lin, Jung-Shan Lin, and Jia-Chin Lin. Channel estimation technique assisted by postfixed pn sequences with zero padding for wireless ofdm communications. In *IWCMC '06 : Proceeding of the 2006 international conference on Communications and mobile computing*, pages 1033–1038, New York, NY, USA, 2006. ACM Press.

[CLLV07] Ray C. C. Cheung, Dong-U Lee, Wayne Luk, and John D. Villasenor. Hardware generation of arbitrary random number distributions from uniform distributions via the inversion method. *IEEE Trans. VLSI Syst*, 15 :952–962, 2007.

[CMB04] J. Chen, J. Moon, and K. Bazargan. Reconfigurable readback-signal generator based on a field-programmable gate array. *IEEE Trans. Magn*, 40 :1744–1750, 2004.

[CN98] Jean-Sébastien Coron and David Naccache. An accurate evaluation of Maurer's universal test. In *SAC '98 : Proceedings of the Selected Areas in Cryptography*, pages 57–71, London, UK, 1998. Springer-Verlag.

[Cor99] Jean-Sébastien Coron. On the security of random sources. *Lecture Notes in Computer Science*, 1560 :29–42, 1999.

[Cor06] Jean-Sebastien Coron. What is cryptography ? *IEEE Security and Privacy*, 04(1) :70–73, 2006.

[CR04] Antonio H. Chan and Gordon W. Roberts. A jitter characterization system using a component-invariant vernier delay line. *IEEE Trans. Very Large Scale Integr. Syst.*, 12(1) :79–95, 2004.

[DG07] Markus Dichtl and Jovan Dj. Golic. High-speed true random number generation with logic gates only. In *Proc. Workshop Cryptographic Hardware and Embedded Systems (CHES'07)*, pages 45–62, 2007.

[DGH07] J.-L. Danger, S. Guilley, and P. Hoogvorst. Fast true random generator, in FPGAs, circuits and systems. In *NEWCAS'07*, 2007.

[DGH09] J.-L. Danger, S. Guilleya, and P. Hoogvorsta. High speed true random number generator based on open loop structures in FPGAs. *Microelectronics Journal*, 2009.

[Dic03] Marcus Dichtl. How to predict the output of a hardware random number generator. In *Proc. Workshop Cryptographic Hardware and Embedded Systems (CHES'03)*, pages 181–188, Springer Berlin / Heidelberg, 2003. Springer-Verlag.

[DVH06] David Delgado, David Vidal, and German Hernandez. Evolutionary design of pseudorandom sequence generators based on cellular automata and its applicability in current cryptosystems. In *GECCO '06 : Proceedings of the 8th annual conference on genetic and evolutionary computation*, pages 1859–1860. ACM Press, 2006.

[DW91] D. DeTemple and S.-H. Wang. Half integer approximations for the partial sums of the harmonic series. *Journal of mathematical analysis and applications*, 160 :149–156, 1991.

[EKN+04] Fung Edmund, Leung Kasto, Parimi Nitin, Pumaprajna Madhura, and GAUDET Vincent. ASIC implementation of a high speed wgng for communication channel emulation. In *IEEE workshop on signal processing systems design and implementation*, 2004.

[EL04] M. D. Ercegovac and T. Lang. *Digital Arithmetic*. Morgan Kaufmann Publishers, 2004.

[Fah05] Amr M. Fahim. *Clock Generators for SOC Processors : Circuits and Architectures (Text, Speech & Language Technology)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[FC09] Claudio Favi and Edoardo Charbon. A 17ps time-to-digital converter implemented in 65nm FPGA technology. In *FPGA '09 : Proceeding of the*

ACM/SIGDA international symposium on Field programmable gate arrays, pages 113–120, New York, NY, USA, 2009. ACM.

[FD03] Viktor Fischer and Milos Drutarovský. True random number generator embedded in reconfigurable hardware. In *Proc. Workshop Cryptographic Hardware and Embedded Systems (CHES'02)*, pages 415–430, London, UK, 2003. Springer-Verlag.

[FDvB04] V. Fischer, M. Drutarovský, M. Šimka, and N. Bochard. High performance true random number generator in Altera Stratix FPLDs. In J. Becker, M. Platzner, and S. Vernalde, editors, *Field-Programmable Logic and Applications – FPL 2004*, volume 3203 of *Lecture Notes in Computer Science*, pages 555–564, Antverp, Belgium, August 30–September 1, 2004. Springer-Verlag.

[FZC04] Yongquan Fan, Zeljko Zilic, and Man Wah Chiang. A versatile high speed bit error rate testing scheme. In *ISQED '04 : Proceedings of the 5th International Symposium on Quality Electronic Design*, pages 395–400, Washington, DC, USA, 2004. IEEE Computer Society.

[GA07] M. George and P. Alfke. Linear feedback shift registers in virtex devices. Technical report, Xilinx Application Note, 2007.

[GLM05] D. Gage, E. Laub, and B. McGarry. Cellular automata : Is rule 30 random ? In *Midwest NKS Conference*, 2005.

[GMV03] Constantinos F. Grecas, Sotirios I. Maniatis, and Iakovos S. Venieris. Introduction of the asymmetric cryptography in GSM, GPRS, UMTS, and its public key infrastructure integration. *Mob. Netw. Appl.*, 8(2) :145–150, 2003.

[Gol06] Jovan Dj. Golić. New methods for digital generation and postprocessing of random data. *IEEE Trans. Comput.*, 55(10) :1217–1229, 2006.

[HG02] Chan A. H. and Roberts G.W. A deep sub-micron timing measurement circuit using a single-stage vernier delay line. In *Custom Integrated Circuits Conference*, 2002.

[HKU03] Akio Hasegawa, Song-Ju Kim, and Ken Umeno. Ip core of statistical test suite of FIPS 140-2. Technical report, Communications Research Laboratory and ChaosWare, Inc., 2003.

[HL00] I. Hatirnaz and Y. Leblebici. A compact modular architecture for the realization of high-speed binary sorting engines based on rank ordering. In *2000 IEEE International Symposium on Circuits and Systems*, 2000. Presented at the 2000 IEEE International Symposium on Circuits and Systems, May 2000, Geneva, Switzerland.

[HMC89] P. D. Hortensius, R. D. McLeod, and H. C. Card. Parallel random number generation for vlsi systems using cellular automata. *IEEE Trans. Comput.*, 38(10) :1466–1473, 1989.

[HMV03] Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[Hua06] Jiun-Lang Huang. On-chip random jitter testing using low tap-count coarse delay lines. *Journal of Electronic Testing*, 22 :387–398, 2006.

[Ist02] Vajda István. Extraction of random bits for cryptographic purposes. *Tatra Mountains Mathematical Publications*, 35 :83–99, 2002.

[Jam80] F James. Monte carlo theory and practice. *Reports on Progress in Physics*, 43(9) :1145–1189, 1980.

[JK99] B. Jun and P. Kocher. The intel random number generator. Technical report, Intel Corporation, 1999.

[KG04] Paul Kohlbrenner and Kris Gaj. An embedded true random number generator for FPGAs. In *FPGA '04 : Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*, pages 71–78, New York, NY, USA, 2004. ACM Press.

[KIKS04] O.O. Khalifa, M.D.R. Islam, S. Khan, and M.S. Shebani. Communications cryptography. In *RF and Microwave Conference*, pages 220 – 223. RFM 2004. Proceedings, Oct 2004.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. *Lecture Notes in Computer Science*, 1666 :388–397, 1999.

[Knu73] D. E. Knuth. *The Art of Computer Programming. Volume 3 : Sorting and Searching.* Addison-Wesley, 1973.

[Knu97] Donald Ervin Knuth. *The art of computer programming, volume 2 : seminumerical algorithms.* Addison-Wesley, Reading, 3rd edition, 1997.

[Kok97] Z. Kokosinski. On parallel generation of combinations in associative processor architectures. In H. El-Rewini, E. Luque, and A. R. Hurson, editors, *Proceedings of the IASTED International Conference on Parallel and Distributed Systems Euro-PDS'97, Barcelona, Spain, June 9-11, 1997*, pages 283–289, Anaheim Calgary Zurich, 1997. ACTA PRESS.

[KS01] Wolfgang Killmann and Werner Schindler. A proposal for : Functionality classes and evaluation methodology for true (physical) random number generators. Technical report, T-Systems debis Systemhaus Information Security Services and Bundesamt für Sicherheit in der Informationstechnik (BSI), 2001.

[KS08] Wolfgang Killmann and Werner Schindler. A design for a physical RNG with robust entropy estimators. In *Proc. Workshop Cryptographic Hardware and Embedded Systems (CHES'08)*, pages 146–163, Berlin, Heidelberg, 2008. Springer-Verlag.

[LAGLM05] Dong-U Lee, Altaf Abdul Gaffar, Wayne Luk, and Oskar Mencer. MiniBit : Bit-width optimization via affine arithmetic. In *DAC'05, Design Automation Conference*, pages 837–840, July 2005.

[LCVL06] D. Lee, R.C.C. Cheung, J.D. Villasenor, and W. Luk. Inversion-based hardware gaussian random number generator : a case study of function evaluation via hierarchical segmentation. In *IEEE International Conference on Field-Programmable Technology (FPT)*, 2006.

[LDH03] Yusuf Leblebici, Tugba Demirci, and Ilhan Hatirnaz. Full-Custom CMOS Realization of a High-Performance Binary Sorting Engine with Linear Area-Time Complexity. In *IEEE International Symposium on Circuits and Systems 2003*, 2003. IEEE International Symposium on Circuits and Systems 2003, Bangkok, May 2003.

[L'E96] P. L'Ecuyer. Combined multiple recursive generators. *Operations Research*, 44 :816–822, 1996.

[Lee06] A hardware gaussian noise generator using the box-muller method and its error analysis. *IEEE Trans. Comput.*, 55(6) :659–671, 2006. Dong-U Lee and John D. Villasenor and Wayne Luk and Philip H. W. Leong.

[LLV+05] Dong-U Lee, W. Luk, J.D. Villasenor, Guanglie Zhang, and P.H.W. Leong. A hardware gaussian noise generator using the wallace method. *IEEE Transactions on*, 911-920 :10, 2005.

[LLVC03a] D.-U Lee, W. Luk, J. Villasenor, and P.Y. K. Cheung. A hardware gaussian noise generator for channel code evaluation. *Field-Programmable Custom Computing Machines, Annual IEEE Symposium on*, 0 :69, 2003.

[LLVC03b] D.-U. Lee, W. Luk, J. Villasenor, and P.Y. K. Cheung. Hierarchical segmentation schemes for function evaluation. In *IEEE Conf. on Field-Programmable Technology*, 2003.

[LLVC04] Dong-U Lee, Wayne Luk, John D. Villasenor, and Peter Y. K. Cheung. A gaussian noise generator for hardware-based simulations. *IEEE Trans. Comput.*, 53(12) :1523–1534, 2004.

[LS07] Pierre L'Ecuyer and Richard Simard. TestU01 : A C library for empirical testing of random number generators. *ACM Trans. Math. Softw.*, 33(4) :22, 2007.

[Mar96] George Marsaglia. Diehard : A battery of tests of randomness. Technical report, Florida State University, Tallahassee, FL, USA, 1996.

[Mar02] Peter Martin. An analysis of random number generators for a hardware implementation of genetic programming using FPGAs and handel-C. Technical Report CSM-358, Department of Computer Science, University of Essex, Wivenhoe Park, Colchester, CO4 3SQ UK., 24th January 2002.

[Mat07] The Mathworks. Matlab curve fitting toolbox, user's guide. Technical report, 2007.

[Mau91] Ueli M. Maurer. A universal statistical test for random bit generators. *Lecture Notes in Computer Science*, 537 :409–420, 1991.

[Max60] J. Max. Quantizing for minimum distortion. *Information Theory, IRE Transactions on*, 6 :7–12, 1960.

[MC94] Edward F. Mykytka and Chun-Yuan Cheng. Generating correlated random variates based on an analogy between correlation and force. In *WSC '94 : Proceedings of the 26th conference on Winter simulation*, pages 1413–1416, San Diego, CA, USA, 1994. Society for Computer Simulation International.

[MCL06] Carlos Roberto Moratelli, Erika Cota, and Marcelo Soares Lubaszewski. A cryptography core tolerant to dfa fault attacks. In *SBCCI '06 : Proceedings of the 19th annual symposium on Integrated circuits and systems design*, pages 190–195, New York, NY, USA, 2006. ACM Press.

[MLR+03] Epstein M., Hars L., Krasinski R., Rosner M., and Zheng H. Design and implementation of a true random number generator based on digital circuit artifacts. In *Proc. Workshop Cryptographic Hardware and Embedded Systems (CHES'03)*, 2003.

[MT00] George Marsaglia and Wai Wan Tsang. The ziggurat method for generating random variables. *Journal of Statistical Software*, 5(8) :1–7, 10 2000.

[MT02] George Marsaglia and Wai Wan Tsang. Some difficult-to-pass tests of randomness. 7(3) :1–8, 2002.

[MTVC06] R. Michard, A. Tisserand, and N. Veyrat-Charvillon. Optimisation d'opérateur arithmétiques matériels à base d'approximations polynomiales. In *SympA*, 2006.

[Mul06] J.-M. Muller. *Elementary Functions, Algorithms and Implementation.* Birkhauser, Boston, 2006.

[MVO96] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography.* CRC Press, Inc., Boca Raton, FL, USA, 1996.

[MZ93] G. Marsaglia and A. Zaman. Monkey tests for random number generators. *Computers Math. Applic.*, 26(9) :1–10, 1993.

[Nat99a] National Institute of Standards and Technology. FIPS PUB 46-3 : Data Encryption Standard (DES). Technical report, 1999.

[Nat99b] National Institute of Standards and Technology. Security requirements for cryptographic modules, FIPS PUB 140-2, 1999.

[Nat01] National Institute of Standards and Technology. Federal information processing standard 197, the advanced encryption standard. Technical report, 2001.

[Nat08] National Institute of Standards and Technology. Recommendation for the triple data encryption algorithm (TDEA) block cipher. Technical report, 2008.

[NKC94] S. Nandi, B. K. Kar, and P. Pal Chaudhuri. Theory and applications of cellular automata in cryptography. *IEEE Trans. Comput.*, 43(12) :1346–1357, 1994.

[NTS99] Noam Nisan and Amnon Ta-Shma. Extracting randomness : A survey and new constructions. *Journal of Computer and System Sciences*, 58 :148–173, 1999.

[OF01] S.F. Oberman and M.J. Flynn. *Advanced Computer Arithmetic Design*. Wiley, 2001.

[Par00] Behrooz Parhami. *Computer Arithmetic : Algorithms and Hardware Designs*. Oxford University Press, 2000.

[PFTV88] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical recipes in C : the art of scientific computing*. Cambridge University Press, New York, NY, USA, 1988.

[PM07] Echeverria P. and Lopez-Vallejo M. FPGA gaussian random number generator based on quintic hermite interpolation inversion. In *IEEE International Midwest Symposium on Circuits and Systems*, 2007.

[Pro00] John Proakis. *Digital Communications*. McGraw-Hill, 2000.

[Rem34] E. Remes. Sur un procédé convergent d'approximations successives pour déterminer les polynômes d'approximation. Technical Report 198, C.R. Acad. Sci. Paris, 1934.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2) :120–126, 1978.

[RSNMS01] A. Rukhin, J. Soto, J. Nechvatal, and D.L. Banks M. Smid. A statistical test suite for random and pseudorandom number generators for statistical applications. *NIST Special Publication in Computer Security*, pages 800–22, 2001.

[Sch01] Werner Schindler. Efficient online tests for true random number generators. In *Proc. Workshop Cryptographic Hardware and Embedded Systems (CHES'01)*, 2001.

[Sch03] Werner Schindler. A stochastical model and its analysis for a physical random number generator. In *IMA Int. Conf.*, pages 276–289, 2003.

[Sch09] Werner Schindler. *Evaluation Criteria for Physical Random Number Generators*, chapter 3, pages 25–54. 2009.

[SFDF06] Martin Simka, Viktor Fischer, Milos Drutarovsky, and Jacques Fayolle. Model of a true random number generator aimed at cryptographic applications. In

In Proceedings of IEEE International Symposium on Circuits and Systems, 2006.

[Sha49] C. E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28 :656–715, 1949.

[Sha02] Ronen Shaltiel. Recent developments in explicit constructions of extractors. In *Bulletin of the EATCS*, 2002.

[SK03] Werner Schindler and Wolfgang Killmann. Evaluation criteria for true (physical) random number generators used in cryptographic applications. In *Proc. Workshop Cryptographic Hardware and Embedded Systems (CHES'02)*, pages 431–449, London, UK, 2003. Springer-Verlag.

[Skl01] Bernard Sklar. *Digital communications : fundamentals and applications.* Prentice-Hall, Inc., 2001.

[SMS07] Berk Sunar, William J. Martin, and Douglas R. Stinson. A provably secure true random number generator with built-in tolerance to active attacks. *IEEE Transactions on Computers*, 56(1) :109–119, 2007.

[SPRQ06] O.-X. Standaert, E. Peeters, G. Rouvroy, and J.-J. Quisquater. An overview of power analysis attacks against field programmable gate arrays. *Proceedings of the IEEE*, 94(2) :383– 394, 2006.

[SPV06] Dries Schellekens, Bart Preneel, and Ingrid Verbauwhede. FPGA vendor agnostic true random number generator. In *Field Programmable Logic and Applications, 2006. FPL '06*, pages 1–6, 2006.

[SR99] Stephen Sunter and Aubin Roy. Bist for phase-locked loops in digital applications. *itc*, 00 :532, 1999.

[SR04] Stephen Sunter and Aubin Roy. On-chip digital jitter measurement, from megahertz to gigahertz. *IEEE Des. Test*, 21(4) :314–321, 2004.

[ST91] Martina F. Schollmeyer and William H. Tranter. Noise generators for the simulation of digital communication systems. In *ANSS '91 : Proceedings of the 24th annual symposium on Simulation*, pages 264–275, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press.

[STCS02] Barry Shackleford, Motoo Tanaka, Richard J. Carter, and Greg Snider. FPGA implementation of neighborhood-of-four cellular automata random number generators. In *FPGA '02 : Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays*, pages 106–112, New York, NY, USA, 2002. ACM Press.

[SV84] Miklos Santha and Umesh V. Vazirani. Generating quasi-random sequences from slightly-random sources (extended abstract). In *FOCS*, pages 434–440, 1984.

[Tau65] R. C. Tausworthe. Random numbers generated by linear recurrence modulo two. *Mathematics of Computation*, 19 :201–209, 1965.

[Tec08] Agilent Technologies. Agilent technologies ezjit and ezjit plus jitter analysis software for infiniium series oscilloscopes. Technical report, 2008.

[Tka02] T. E. Tkacik. A hardware random number generator. In *Proc. Workshop Cryptographic Hardware and Embedded Systems (CHES'02)*, 2002.

[TL07] D.B. Thomas and W. Luk. Non-uniform random number generation through piecewise linear approximations. *IET Computers & Digital Techniques*, 1 :1312–321, 2007.

[TLL03] K. H. Tsoi, K. H. Leung, and P. H. W. Leong. Compact FPGA-based true and pseudo random number generators. In *FCCM '03 : Proceedings of the 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, page 51, Washington, DC, USA, 2003. IEEE Computer Society.

[TSP00] Marco Tomassini, Moshe Sipper, and Mathieu Perrenoud. On the generation of high-quality random numbers by two-dimensional cellular automata. *IEEE Trans. Comput.*, 49(10) :1146–1151, 2000.

[VHKK08] Ihor Vasyltsov, Eduard Hambardzumyan, Young-Sik Kim, and Bohdan Karpinskyy. Fast digital TRNG based on metastable ring oscillator. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Proc. Workshop Cryptographic Hardware and Embedded Systems (CHES'08)*, volume 5154 of *Lecture Notes in Computer Science*, pages 164–180. Springer, 2008.

[Vil07] Mark B. Villarino. Ramanujan's harmonic number expansion into negative powers of a triangular number. *Journal of inequalities in pure and applied mathematics*, 2007.

[VY03] Branka Vucetic and Jinhong Yuan. *Space-Time Coding*. John Wiley & Sons, 2003.

[Wal96] C. S. Wallace. Fast pseudorandom generators for normal and exponential variates. *ACM Trans. Math. Softw.*, 22(1) :119–127, 1996.

[Wol86] Stephen Wolfram. Random sequence generation by cellular automata. *Adv. Appl. Math.*, 7(2) :123–169, 1986.

[Wol02] Stephen Wolfram. *A new kind of science*. 2002.

[Xil02] Xilinx. Superior jitter management with DLLs. Technical Report VTT013 (v1.2), Xilinx, 2002.

[Xil08] Xilinx. Virtex-5 FPGA configuration user guide. Technical report, 2008.

[Xil09] Xilinx. Divider generator v3.0 ds530. Technical report, 2009.

[XJMF04] Zhao Xuelong, Wang Jiwen, Xu Manwu, and Liu Fengyu. High-quality pseudo-random sequence generator based on one-dimensional extended cellular automata. In *InfoSecu '04 : Proceedings of the 3rd international conference on Information security*, pages 222–223, New York, NY, USA, 2004. ACM Press.

[YSKB] Sang-Kyung Yoo, Berk Sunar, Deniz Karakoyunlu, and Berk Birand. A robust and practical random number generator.

[Zim] Reto Zimmermann. Computer arithmetic : Principles, architectures, and VLSI design. Technical report.

[ZLL+05] Guanglie Zhang, Philip Heng Wai Leong, Dong-U Lee, John D. Villasenor, Ray C. C. Cheung, and Wayne Luk. Ziggurat-based hardware gaussian random number generator. In *FPL*, pages 275–280, 2005.

Publications

Conférences internationales

[1] Renaud Santoro. Accelerating statistical tests for real-time estimation of randomness. In *Note : 5th International Workshop on Cryptographic Architectures Embedded in Reconfigurable Devices*, 2007.

[2] Renaud Santoro. Evaluation of TRNGs under various experimental conditions. In *Note : 7th International Workshop on Cryptographic Architectures Embedded in Reconfigurable Devices*, 2009.

[3] Renaud Santoro, Sébastien Roy, and Olivier Sentieys. Search for optimal five-neighbor FPGA-based cellular automata random number generators. In *Signals, Systems and Electronics, 2007. ISSSE*, 2007.

[4] Renaud Santoro, Olivier Sentieys, and Sébastien Roy. On-line monitoring of random number generators for embedded security. In *IEEE International Symposium on circuit and Systems, ISCAS*, 2009.

[5] Renaud Santoro, Olivier Sentieys, and Sébastien Roy. On-the-fly evaluation of FPGA-based true random number generator. In *IEEE Computer Society Annual Symposium on VLSI, ISVLSI*, 2009.

[6] Renaud Santoro, Arnaud Tisserand, Olivier Sentieys, and Sebastien Roy. Arithmetic operators for on-the-fly evaluation of TRNGs. volume 7444. SPIE, 2009.

Conférence nationale

[1] Renaud Santoro. Évaluation de trng dans diverses conditions expérimentales. In *Crypto'Puces*, 2009.

Table des figures

0-1 Présentation du générateur proposé. Le circuit, implanté au sein d'un FPGA, est composé d'un TRNG et d'un PRNG. La séquence aléatoire produite par le TRNG est analysée en temps réel, par des mécanismes de test en ligne. En cas de défaillances importantes du générateur, une alarme est levée et les séquences déficientes sont supprimées. L'analyse permet d'assurer un vecteur d'initialisation très aléatoire. 3

0-2 Système de communications numériques avec processus de compression des données et de protection contre les erreurs de transmission. 5

1-1 Générateur aléatoire hybride composé d'un TRNG et d'un PRNG. 8

1-2 Décomposition d'un générateur vraiment aléatoire. Un TRNG peut être décomposé en trois parties, le bruit physique, un échantillonneur et un module de post-traitement 16

1-3 Schéma de principe du TRNG [BB99]. Les principaux blocs composant le TRNG sont une source de bruit physique externe, un étage de filtrage et un comparateur à hystérésis. 17

1-4 Schéma de principe du TRNG [JK99]. Le bruit thermique du générateur est obtenu en soustrayant les signaux relevés en sortie de deux résistances. La différence des deux signaux impliquant un bruit thermique est ensuite amplifiée. Le signal obtenu est appliqué en entrée d'un oscillateur contôlé en tension (VCO) puis échantillonnée par l'horloge Clk. 18

1-5 Fonctionnement d'un OS selon [TLL03]. Le principe est de réaliser l'échantillonnage d'une horloge haute fréquence (F_h) par une horloge basse fréquence (F_l). 18

1-6 Implantation de l'OS selon [TLL03]. Décorrélateur utilisé par le générateur. 19

1-7 Implantation de l'OS à partir de deux PLL, selon [FDvB04]. Le principe de fonctionnement du générateur est de traquer la gigue relative entre deux signaux d'horloges. Les deux signaux d'horloges sont générés à partir de blocs PLL. La variation de bruit dans une PLL engendre une variation de la fréquence du signal généré. Cette variation représente la gigue. 20

1-8 Observation d'un échantillonnage critique dans [FDvB04]. 21

1-9 Mise en place de la ligne de retard dans [FDvB04]. Echantillonnage du signal CLJ retardé par plusieurs délais puis combinaison des sorties par une porte XOR. 22

1-10 Principe de fonctionnement de l'OS selon [BGL+03]. L'oscillateur rapide (CLK_{FAST}) est implanté par un RO comportant 10 inverseurs, sa fréquence est de 1 GHz. L'horloge CLK_{JIT} possède une fréquence moyenne de 10 MHz. Le post-traitement permet de contrôler le facteur N_{PRE} afin d'adapter la fréquence du signal CLK_{RNG}. 23

1-11 Signal en sortie des oscillateurs selon [BGL+03]. 24

1-12 Schéma de principe du TRNG selon [KG04]. Deux horloges sont fournies par deux RO. Les périodes des deux horloges sont légèrement différentes. . . 25

1-13 Schéma de l'oscillateur à anneau selon [KG04]. 25

1-14 Schéma de principe du bloc échantillonneur de [KG04]. Les signaux $R0$, $S0$ et $E0$ sont des signaux de contrôle. Le circuit permet d'extraire la gigue présente sur les deux horloges. 26

1-15 Chronogramme de fonctionnement de l'échantillonneur de [KG04]. 26

1-16 Schéma du générateur hybride de [Tka02]. Le TRNG est composé de deux sources aléatoires (deux RO) et de deux générateurs pseudo-aléatoires, un LFSR et un automate cellulaire). 28

1-17 Schéma de principe du TRNG de [SMS07]. Le TRNG est constitué de 114 RO composés de 13 inverseurs. Combinaison des sorties des RO par un arbre de XOR. Une fonction résiliente (*resilient function*) est choisie puis réalisée en logiciel à partir d'un code BCH (256,16,113) pour décorréler la signal DAS. 29

1-18 Schéma de principe du TRNG de [BGL+06]. Les deux horloges sont générées à partir de deux RO. L'aléa produit par le TRNG est engendré par la gigue relative entre les deux horloges. Lorsque la gigue relative est supérieure à un certain seuil, un bit aléatoire est produit en sortie du TRNG. 30

1-19 Schéma de principe du TRNG de [Gol06]. Le TRNG est composé d'un oscillateurs à anneau de Fibonacci (FIRO), d'un oscillateurs à anneau de Galois (GARO), d'un RO classique et d'un module de post-traitement (un LFSR). 31

1-20 Schéma de principe de l'oscillateur à anneau de Fibonacci selon [Gol06]. . . 31

1-21 Schéma de principe de l'oscillateur à anneau de Galois selon [Gol06]. 31

1-22 TRNG basé sur l'utilisation de GARO dans [DG07]. Le TRNG est basé sur l'utilisation d'un GARO. Un compteur modulo deux compte le nombre de transitions 0-1 effectuées par le signal oscillant du GARO pendant un cycle de l'horloge CLK. La sortie du GARO est échantillonnée par une bascule D. Le TRNG ne nécessite aucun module de post-traitement. 32

1-23 Schéma de principe du TRNG de [MLR+03]. Le TRNG est composé de deux multiplexeurs et de deux inverseurs. Le TRNG fonctionne suivant deux phases, une phase d'oscillation et une seconde phase de métastabilité. Si l'entrée *Select* est égale à 0, le TRNG se résume à deux RO classiques. Lorsque *Select* vaut 1, les inverseurs sont mis en cascade et le circuit est stabilisé. La métastabilité du circuit est rencontrée lors d'une transition 0-1 sur le signal *Select*. 33

1-24 Principe du TRNG proposé par Danger et al. dans [DGH09]. 34

1-25 TRNG proposé par Danger et al. dans [DGH09]. 35

1-26 TRNG entier proposé par Danger et al. dans [DGH09]. La chaîne de délai correspond au schéma présenté à la figure 1-25. En sortie, un post-traitement est utilisé afin de diminuer le biais de la séquence générée. [DGH09] utilise un décorrélateur de Von-Neumann. 35

1-27 Architecture META-RO proposée par Vasyltsov et al. dans [VHKK08]. Le TRNG est basé sur la possibilité de mettre chaque inverseur dans un état de métastabilité lorsque le signal CLK est égal à 0. Ensuite, lorsque le signal CLK est égal à 1, l'architecture est similaire à un RO classique. 36

1-28 TRNG utilisant le circuit META-RO de [VHKK08]. Lors de l'étape de métastabilité (appelée MS), les multiplexeurs déconnectent les inverseurs les uns des autres. Cette étape met chaque inverseur dans un état métastable. Ensuite, dans la seconde étape, les inverseurs sont reconnectés entre eux et forment un oscillateur à anneau classique. Un signal oscillant est produit en sortie de chaque inverseur. Pendant cette étape, le signal oscillant est échantillonné par le signal d'horloge CLK'. 36

2-1 TRNG proposé par Sunar et al. [SMS07]. Le TRNG est composé de 114 oscillateurs à anneau possédant 13 inverseurs. Un arbre de XOR est utilisé afin de combiner les sorties des oscillateurs. Le bloc de post-traitement est effectué par un code BCH [256,16,113]. 44

2-2 Architecture META-RO proposée par Vasyltsov et al. dans [VHKK08]. Le TRNG est basé sur la possibilité de mettre chaque inverseur dans un état de métastabilité lorsque le signal CLK est égal à 0. Ensuite, lorsque le signal CLK est égal à 1, l'architecture est similaire à un RO classique. 45

2-3 TRNG utilisant le circuit META-RO de [VHKK08]. Lors de l'étape de métastabilité (appelée MS), les multiplexeurs déconnectent les inverseurs les uns des autres. Cette étape met chaque inverseur dans un état métastable. Ensuite, dans la seconde étape, les inverseurs sont reconnectés entre eux et forment un oscillateur à anneau classique. Un signal oscillant est produit en sortie de chaque inverseur. Pendant cette étape, le signal oscillant est échantillonné par le signal d'horloge CLK'. 45

2-4 Schéma de principe de l'oscillateur à anneau de Galois selon [Gol06]. 45

2-5 TRNG basé sur l'utilisation de GARO dans [DG07]. Le TRNG est basé sur l'utilisation d'un GARO. Un compteur modulo deux compte le nombre de transitions 0-1 effectuées par le signal oscillant du GARO pendant un cycle de l'horloge CLK. La sortie du GARO est échantillonnée par une bascule D. Le TRNG ne nécessite aucun module de post-traitement. 46

2-6 Simulation d'une activité autour de la source de bruit des TRNG en plaçant un grand nombre d'oscillateurs à anneau. 48

2-7 La méthodologie proposée est d'évaluer la qualité des TRNG grâce à l'utilisation de tests statistiques embarqués, fonctionnant en temps réel sur l'ensemble de la séquence. 49

2-8 Taux de réussite des tests du FIPS 140-2 sans activité autour du générateur de Sunar et al. [SMS07]. 53

2-9 Taux de réussite des tests du FIPS 140-2 avec activité autour du générateur de Sunar et al. [SMS07]. 53

2-10 Taux de réussite des tests du FIPS 140-2 sans activité autour du générateur de Vasyltsov et al. [VHKK08]. 53

2-11 Taux de réussite des tests du FIPS 140-2 avec activité autour du générateur de Vasyltsov et al. [VHKK08]. 53

2-12 Taux de réussite des tests du FIPS 140-2 sans activité autour du générateur de Dichtl et al. [DG07]. 54

2-13 Taux de réussite des tests du FIPS 140-2 avec activité autour du générateur de Dichtl et al. [DG07]. 54

2-14 Analyse de gigue cycle à cycle d'un RO implanté au sein d'un FPGA Altera EP2S60 porté à une température de 25°C. 56

2-15 Analyse de gigue cycle à cycle d'un RO implanté au sein d'un FPGA Xilinx V4SX35 porté à une température de 25°C. 57

2-16 Analyse de gigue cycle à cycle d'un RO soumis à une forte activité parasite implanté au sein d'un FPGA Altera EP2S60 porté à une température de 25°C. 58

2-17 Analyse de gigue cycle à cycle d'un RO soumis à une forte activité parasite implanté au sein d'un FPGA Xilinx V4SX35 porté à une température de 25°C. 59

2-18 La solution proposée pour optimiser le débit des TRNG est d'utiliser un générateur d'horloge reconfigurable dynamiquement. Celle-ci repose sur l'utilisation des tests statistiques décrits par l'AIS 31 et ceux du FIPS 140-2. Les tests sont sélectionnés comme métrique afin d'évaluer la qualité des TRNG. 60

2-19 Tri en série suivant l'ordre croissant des 2^{16} mots de 48 bits. 61

2-20 Exemple de tri pendant la phase d'acquisition des 2^{16} mots de 48 bits. Lors de la phase d'acquisition, à chaque réception d'un nouveau mot, les bancs déterminent la valeur maximale entre la valeur en entrée et celle des deux registres. 62

2-21 Une fois les 2^{16} mots reçus, la seconde phase de tri commence. Les bancs de registres fonctionnent dans le sens inverse. La direction des bus entre les bancs est inversée. Durant 2^{16} cycles, les bancs évaluent la valeur minimale entre le contenu des registres et l'entrée. La valeur minimale est ensuite transférée en entrée du banc suivant. 62

2-22 Graphe flot de signal nécessaire à la réalisation du test T7a. 65

2-23 Fonction $H'''(i)$ à approcher pour $i \in]0,1[$. L'asymptote verticale pour les valeurs faibles de i rend l'approximation difficile. 69

2-24 Segmentation de niveau 0 pour $H'''(i)$ où $i \in [2^{-18},1[$. 71

2-25 Précision de l'approximation (en bits corrects) pour $i \in [2^{-18},1[$. 72

2-26 Taille mémoire en fonction de la méthode utilisée pour approcher la fonction $H'''(i)$ pour $i \in [2^{-18},2^{-18+j}[$. 72

2-27 Graph flot signal réalisant l'approximation de $H'''(i)$ pour $i \in [2^{-13},1[$. . . . 73

2-28 Précision de l'approximation (en bits corrects) de $H'''(i)$ pour $i \in [2^{-13},1[$. . 74

2-29 *Leading Zero Detector* (LZD). Des opérateurs OU logiques appliqués en cascade sur les 13 bits de poids forts et un compteur de 1 sont requis. Le LZD produit la variable *P2SR address* pouvant prendre une valeur entre 1 et 13. 74

2-30 Organisation de la ROM pour la mémorisation du coefficient p_0. 75

2-31 Pourcentage des quatre classes lorsque le TRNG Sunar et al. [SMS07] est analysé par l'ensemble des tests T_{r} tests pour différents débits de génération allant de 1.03 Mb/s à 10.65Mb/s avec des pas de 343.7 Kb/s. L'axe des abscisses est ici exprimé en Mb/s. 78

2-32 Pourcentage des quatre classes lorsque le TRNG Dichtl et al. [DG07] est analysé par l'ensemble des tests T_{r} tests pour différents débits de génération allant de 1.03 Mb/s à 10.65Mb/s avec des pas de 343.7 Kb/s. L'axe des abscisses est ici exprimé en Mb/s. 78

2-33 Pourcentage des quatre classes lorsque le TRNG Sunar et al. [SMS07] est analysé par l'ensemble des tests T_{w} tests pour différents débits de génération allant de 1.03 Mb/s à 10.65Mb/s avec des pas de 343.7 Kb/s. L'axe des abscisses est ici exprimé en Mb/s. 79

2-34 Pourcentage des quatre classes lorsque le TRNG Dichtl et al. [DG07] est analysé par l'ensemble des tests T_{w} pour différents débits de génération allant de 1.03 Mb/s à 10.65Mb/s avec des pas de 343.7 Kb/s. L'axe des abscisses est ici exprimé en Mb/s. 79

2-35 Présentation d'un RNG hybride disposant de mécanisme de test en ligne de la qualité du TRNG. À partir de l'évaluation de la qualité du signal DAS et de la séquence binaire produite après post-traitement, le comportement du TRNG est analysé. En cas de défaillances importantes du TRNG, une alarme est levée. L'analyse, en temps réel de la qualité du TRNG permet d'assurer un excellent germe au PRNG. 82

3-1 RNG hybride développé conjointement avec l'entreprise spécialisée en cryptographie. Le générateur est composé du TRNG proposé par Sunar et al. [SMS07]. La sortie du TRNG est évaluée en temps réel par la batterie de tests matériels FIPS 140-2. Le germe est ensuite utilisé pour initialiser un PRNG, ici un AES-128. 84

3-2 Présentation du circuit OCHREv1. 86

3-3 Schéma layout du circuit OCHREv1. 87

3-4 Présentation des caractéristiques de la gigue cycle à cycle de l'oscillateur à anneau possédant 33 inverseurs. La mesure est effectuée à partir de l'observation de 10^6 points échantillonnés à 20×10^6 éch/s. L'horloge analysée possède une fréquence de 216.96 MHz. L'histogramme montre la distribution de la gigue. En abscisse, l'échelle est égale à 50 ps par division. La valeur moyenne (μ) de la gigue cycle à cycle est de -122.3 fs. Les valeurs maximale et minimale sont respectivement de 98.3 ps et -86.6 ps. L'écart type est de 30.87 ps. Pour le spectre de puissance, l'échelle en ordonnée est de 12 ps par division et de 10.8 MHz par division en abscisse. 89

3-5 Présentation des caractéristiques de la gigue cycle à cycle de l'oscillateur à anneau possédant 33 inverseurs. La mesure est effectuée à partir de l'observation de 10^6 points échantillonnés à 20×10^6 éch/s. La fréquence de l'horloge analysée est de 79.99 MHz. L'échelle en abscisse de l'histogramme est de 5 ps par division. La valeur moyenne et l'écart type sont respectivement de 85.8 ps et 4.62 ps. Par rapport au RO à 33 inverseurs dans lequel la gigue moyenne est proche de 0s (-112.3 fs) et l'écart type élevé (30.87 ps), ici la gigue observée est relativement faible et non centrée autour de 0. Les valeurs maximale et minimale de la gigue sont respectivement de 104.1 ps et 71.38 ps. Pour le spectre de puissance, l'échelle en ordonnée est de 1.25 ps par division et de 4 MHz par division en abscisse. 90

3-6 Visualisation du bruit échantillonné (signal *outRingOscillator*) en sortie du TRNG Schellekens et al. 91

3-7 Le circuit est composé des TRNG proposés par Sunar et al. [SMS07], Vasyltsov et al. [VHKK08], Dichtl et al. [DG07] et Yoo et al. [YSKB]. Les tests statistiques AIS 31 et FIPS 140-2 sont implantés. La séquence issue du TRNG est validée par le FIPS 140-2 et sert ensuite à initialiser un générateur pseudo-aléatoire, un AES-128 fonctionnant en mode OFB. En sortie de l'AES, deux blocs sont implantés afin de générer des variables aléatoires suivant une loi de Rayleigh et une loi gaussienne (via l'unité Box-Muller). 93

3-8 Schéma layout du circuit OCHREv2. 95

3-9 Système de communication numérique avec processus de compression des données et de protection contre les erreurs de transmission. 96

3-10 Fonction $f(u_0)$. 106

3-11 Segmentation hybride [AFCS08]. 107

3-12 Dérivée de la fonction $f(u_0)$. Celle-ci montre la pente de la fonction f. Lorsque les valeurs de u_0 sont proches de 0 ou de 1, le coefficient directeur est très élevé. 108

3-13 Fonction $f(u_0)$ pour $u_0 \in [0.25, 0.875[$. 109

3-14 Fonction $f(u_0)$ pour $u_0 \in [2^{-32}, 2^{-2}[$. 110

3-15 Précision obtenue lors l'approximation de $f(u_0)$ pour $u_0 \in [2^{-16}, 2^{-2}[$ avec des polynômes de degrés 2 et 3. 111

3-16 Coût mémoire total (en bits) lorsque la fonction $f(u_0)$ est approcher pour $u_0 \in [2^j, 2^{-2}[$ et mémorisée pour $u_0 \in [2^{-16}, 2^j[$ pour $j \in \{-14, \ldots, -3\}$. . . 112

3-17 Fonction $f(u_0)$ pour $u_0 \in [0.875; 1[$. 113

3-18 Schéma de segmentation et précision de l'approximation de $f(u_0)$ pour $u_0 \in [0.875, 1[$. 114

3-19 Coût mémoire total (en bits) engendré par une approximation réalisée sur les j premiers intervalles. 114

3-20 Graphe flot de signal réalisant le calcul de $f(u_0)$. 115

3-21 Fonction $g_0(u_1)$. 116

3-22 Graphe flot de signal réalisant le calcul de $g_0(u_1)$. 117

3-23 Représentation de 10^4 couples de variables aléatoires générées par le GRNG. 119

3-24 Histogramme de la variable aléatoire gaussienne. 119

4-1 Schéma de principe d'un automate cellulaire [Abd03]. Un automate est basé sur la connexion locale de N cellules identiques. 124

4-2 Règle 30 [Wol02]. 124

4-3 Évolution de l'état des cellules à partir de la règle 30, [Wol02]. L'évolution est basée sur l'initialisation de la cellule centrale à "1" et le reste des cellules à "0". Une ligne correspond à un cycle du générateur. 125

4-4 Automate cellulaire mono-dimensionnel. Le schéma de connexion, ici $\{-1, 0, 1, 2\}$, définit les relations entre les cellules. 126

4-5 Automate cellulaire bi-dimensionnel. Le schéma de connexion, ici $\{2n2o, c, ne, s\}$, définit les relations entre les cellules. Les notations n, e, s et o désignent les points cardinaux Nord, Est, Sud, Ouest. L'écriture $2n2o$ désigne une cellule située deux positions vers le Nord et deux position vers l'Est. Le c désigne la cellule courante. 126

4-6 Automate cellulaire tri-dimensionnel. Le schéma de connexion, ici $\{dno, c, dne, d\}$, définit les relations entre les cellules. Les notations n, e, s et o désignent les points cardinaux Nord, Est, Sud, Ouest. Les notations d et u sont utilisées pour indexer la troisième dimension de l'automate. Le c désigne la cellule courante. 126

4-7 Exemple d'écart spatial avec $\gamma = 3$. 127

4-8 La figure présente le principe du calcul de (4-2). Une mémoire accessible par son contenu (CAM) est utilisée afin de mémoriser le nombre d'occurrences des blocs 8 bits. 130

4-9 Fonction $p_i \log_2 p_i$ pour $p_i \in \{\frac{1}{2.5\times10^3}, \ldots, 1\}$. 130

4-10 Fonction $p_i \log_2 p_i$ pour $p_i \in \{\frac{1}{2.5\times10^3}, \ldots, \frac{40}{2.5\times10^3}\}$. 130

4-11 La figure présente l'architecture de la recherche exhaustive. Celle-ci utilise N_b batteries en parallèle. Les automates cellulaires sont implantés à partir de blocs RAM. Il est possible de reconfigurer facilement les règles des CA. L'architecture est implantée dans un FPGA Virtex 5 LX 50. La fréquence du circuit est de 188.83 MHz. La recherche exhaustive est réalisée en 3 heures et 32 minutes. 131

4-12 La figure compare l'entropie 8 bits du CA optimal à 5 voisins trouvé, d'un CA30 [Wol86] à 64 cellules,d'un LFSR de longueur 64 (LFSR64) [GA07] et du CA1D à 4 entrées de longueur 64 (règle $CA639_{16}$) [STCS02]. Les lignes représentées par des signes $+$, $\circ$ et $\square$ correspondent respectivement au calcul de l'entropie pour une séquence de 1×10^4, 2×10^4 et 2×10^5 bits aléatoires. Le CA à 5 entrées converge plus rapidement vers l'entropie maximale. . . . 132

4-13 Répartition en deux dimensions d'une séquence aléatoire de 400 entiers 32 bits générés par le CA à 5 voisins. 133

4-14 Répartition en deux dimensions d'une séquence aléatoire de 400 entiers 32 bits générés par le CA à 4 voisins. 133

4-15 Répartition en deux dimensions d'une séquence aléatoire de 400 entiers 32 bits générés par le CA30. 134

4-16 Répartition en deux dimensions d'une séquence aléatoire de 400 entiers 32 bits générés par le LFSR64. 134

4-17 Répartition en trois dimensions d'une séquence aléatoire de 400 entiers 32 bits générés par le CA à 5 voisins. 134

4-18 Répartition en trois dimensions d'une séquence aléatoire de 400 entiers 32 bits générés par le CA à 4 voisins. 134

4-19 Répartition en trois dimensions d'une séquence aléatoire de 400 entiers 32 bits générés par le CA30. 134

4-20 Répartition en trois dimensions d'une séquence aléatoire de 400 entiers 32 bits générés par le LFSR64. 134

4-21 Répartition en deux dimensions d'une séquence aléatoire de 2×10^4 entiers 32 bits générés par le CA à 5 voisins. 135

4-22 Répartition en deux dimensions d'une séquence aléatoire de 2×10^4 entiers 32 bits générés par le CA à 4 voisins. 135

4-23 Répartition en deux dimensions d'une séquence aléatoire de 2×10^4 entiers 32 bits générés par le CA30. 135

4-24 Répartition en deux dimensions d'une séquence aléatoire de 2×10^4 entiers 32 bits générés par le LFSR64. 135

4-25 Répartition en trois dimensions d'une séquence aléatoire de 2×10^4 entiers 32 bits générés par le CA à 5 voisins. 135

4-26 Répartition en trois dimensions d'une séquence aléatoire de 2×10^4 entiers 32 bits générés par le CA à 4 voisins. 135

4-27 Répartition en trois dimensions d'une séquence aléatoire de 2×10^4 entiers 32 bits générés par le CA30. 136

4-28 Répartition en trois dimensions d'une séquence aléatoire de 2×10^4 entiers 32 bits générés par le LFSR64. 136

4-29 Estimation de la fonction d'autocorrélation du CA à 5 voisins. 136

4-30 Estimation de la fonction d'autocorrélation du CA à 4 voisins. 136

4-31 Estimation de la fonction d'autocorrélation du CA30. 137

4-32 Estimation de la fonction d'autocorrélation du LFSR64. 137

4-33 Architecture pour le tri en série selon [BC98]. 140

4-34 Graphe flot de signal du test du χ^2 dans le *birthday test*. 141

Liste des tableaux

1-1 Comparaison de la performance de TRNG. Dans la seconde colonne, le tableau résume le type de source de bruit utilisé. La colonne trois évalue la possibilité de synthétiser entièrement le TRNG au sein d'un circuit VLSI ou FPGA. Enfin, la colonne quatre présente le débit du générateur. Lorsqu'une information est indisponible dans la littérature, la notation Inc. est reportée dans le tableau. 42

2-1 Intervalles requis dans le *run* test. 50

2-2 Surface totale, puissance dynamique consommée et chemin critique en utilisant une technologie CMOS 130 nm, 1.2V. 51

2-3 Résultat de l'implantation du FIPS 140-2 dans les FPGA Xilinx ciblés. . . . 51

2-4 Résultat de l'implantation du FIPS 140-2 dans le FPGA Altera Stratix II ciblé. 52

2-5 Intervalles requis dans le *run* test. 63

2-6 Segmentation P2SR pour une entrée 8 bits, $v_0 = 5$ et $v1 = 3$. 71

2-7 Résultats d'implantation pour une cible FPGA Xilinx Virtex 5 XC5VSX50. 75

2-8 Résultat de l'implantation des tests de l'AIS 31 pour une cible FPGA Xilinx Virtex 5 SX50T-1. 76

2-9 Nombre d'ALUT requis par les TRNG au sein du FPGA Altera EP2S60. . 80

2-10 Puissance dynamique consommée par les trois générateurs implantés au sein de la carte de développement FPGA Actel Igloo. 81

3-1 Surface totale, puissance dynamique consommée et chemin critique pour une technologie CMOS 130 nm, 1.2V. 86

3-2 Répartition de la gigue cycle à cycle entre $\mu \pm 1\sigma$, $\mu \pm 2\sigma$ et $\mu \pm 3\sigma$. 88

3-3 Résumé de la réussite (en %) du TRNG Schellekens et al. obtenue aux tests FIPS 140-2. Les résultats proviennent de l'analyse de 1M séquences binaires composées de 2×10^4 bits. 91

3-4 Fréquences maximales des blocs du circuit OCHREv2 synthétisés et placés-routés en technologie CMOS 130 nm. 94

3-5 Surface et consommation dynamique des blocs du circuit OCHREv2 synthétisés et placés-routés en technologie CMOS 130 nm. 94

3-6 Comparaison de la performance de GRNG utilisant la méthode BM. 100

3-7 Comparaison des techniques de GRNG. 105

3-8 Précision minimale de $p^\star$ pour $u_0 \in [0.25, 0.875[$ lorsque le polynôme est de degré 1. 109

3-9 Précision minimale de $p^\star$ pour $u_0 \in [0.25, 0.875[$ lorsque le polynôme est de degré 2. 109

3-10 Précision minimale de $p^\star$ pour $u_0 \in [0.25, 0.875]$ lorsque le polynôme est de degré 3. 110

3-11 Représentation en virgule fixe des coefficients et coût mémoire en bits avec et sans réduction de l'intervalle. 110

3-12 Représentation en virgule fixe des coefficients. 112

3-13 Représentation en virgule fixe des coefficients. 115

3-14 Coût mémoire (en bits) engendré par l'approximation polynômiale et par une mémorisation directe des valeurs de $f(u_0)$ sur les cinq segments. 115

3-15 Nombre de segments requis, nombre de bits consacrés à la partie fractionnaire des coefficients, précision minimale de l'approximation et coût mémoire engendré par des polynômes de degrés 2 et 3. 116

3-16 Résultats d'implantation pour une cible FPGA Xilinx Virtex 5 SX50. Présentation du nombre total de LUT, de la fréquence maximale, du nombre de BRAM et de DSP48. De plus, la largeur du mot en sortie et le débit du générateur sont présentés. 117

3-17 Résultats d'implantation pour une cible FPGA Xilinx Virtex 5 SX50. 118

3-18 Comparaison de la performance de l'architecture présentée avec les GRNG utilisant la méthode BM. 118

4-1 Résultat de l'implantation de l'algorithme produisant les C_{32}^{16} combinaisons possibles. 129

4-2 Le tableau résume les valeurs d'entropie obtenues lorsqu'une séquence de 2×10^5 bits est analysée et montre que le CA à 5 entrées possède l'entropie la plus élevée, 7.9942 bits. De plus, celle-ci est très proche de l'entropie maximale $H_{max} = 8$. 133

4-3 Résultat de l'implantation du bloc de tri dans un FPGA Xilinx Virtex 5 SX50. 140

4-4 Résultat de l'implantation du *gcd test* dans un FPGA Xilinx Virtex 5 SX50. 144

Glossaire

3DES *Triple DES*, 9

AES *Advanced Encryption Standard*, 9

ASIC *Application-Specific Integrated Circuit*, 3

BCH Code correcteur de nom des inventeurs Bose, Ray-Chaudhuri et Hocquenghem, 26

BSI *Bundesamtes für Sicherheit in der Informationstechnik*, 37

CA *Cellular Automata*, 119

CLB *Configurable Logic Block*, 25

CMOS *Complementary Metal Oxide Semiconductor*, 11

CPA *Correlation Power Analysis*, 11

DAS *Digitized Analog Signal*, 14

DCSSI Direction centrale de la sécurité des systèmes d'Information, 80

DES *Data Encryption Standard*, 9

DFA *Differential Fault Analysis*, 10

DPA *Differential Power Analysis*, 10

ECC *Elliptic Curve Cryptosystems*, 10

FIPS *Federal Information Processing Standard*, 36

FIRO *Fibonacci Ring Oscillator*, 29

FPGA *Field Programmable Gate Array*, 1

GARO *Galois Ring Oscillator*, 29

GCD *Greatest Common Divisor*, 18

GF *Galois Field*, 29

GPRS *General Packet Radio Service*, 10

GRNG *Gaussian Random Number Generator*, 93

GSM *Global System for Mobile Communications*, 10

IA *Invasive Attacks*, 10

LFSR *Linear Feedback Shift Register*, 120

MIMO *Multiple-Input Multiple-Output*, 13

NIA *Non-Invasive Attacks*, 10

NIST *National Institute of Standards and Technology*, 36

OCHRE *On-CHip Randomness Evaluation*, 81

OFB *Output Feedback Mode* , 89

OS *Oscillator Sampling*, 16

PLL *Phase-Locked Loop*, 17

PRNG *Pseudo Random Number Generator*, 1

QMC Méthodes de Quasi Monte-Carlo, 12

RNG *Random Number Generator*, 1

RO *Ring Oscillator*, 21

RSA Ron Rivest, Adi Shamir et Len Adleman, 10

SHA *Secure Hash Algorithm*, 16

SNR *Signal Noise Ratio*, 13

SOC *System On-Chip*, 1

TCL Théorème Centrale Limite, 95

TRNG *True Random Number Generator*, 1

UMTS *Universal Mobile Telecommunications System*, 10

v.a. Variable aléatoire, 12

VCO *Voltage-Controlled Oscillator*, 15

VDL *Vernier Delay Line*, 151

VLSI *Very-Large-Scale Integration*, 1

Résumé

L'objectif d'un générateur de nombres aléatoires (RNG) est de produire une séquence binaire aléatoire indépendante, imprédictible et uniformément répartie. Les RNG sont nécessaires dans toutes sortes d'applications telles que la cryptographie, les télécommunications, les simulations numériques, le test de circuits VLSI ou encore les algorithmes probabilistes. Ainsi, le nombre d'applications embarquées nécessitant des RNG matériels augmente constamment, spécialement au sein des circuits reconfigurables. En vue de la conception matérielle d'un RNG, la solution est d'utiliser un générateur hybride composé d'un générateur vraiment aléatoire (TRNG) et d'un générateur pseudo-aléatoire (PRNG). Prouver mathématiquement qu'un générateur est aléatoire est généralement difficile, voire impossible. Les RNG sont souvent évalués via des tests statistiques. L'évaluation d'un TRNG est beaucoup plus complexe que celle d'un PRNG car la qualité d'un TRNG est fonction de la source de bruit utilisée, de l'environnement extérieur et de la qualité d'implantation au sein du circuit ciblé. Fischer et al. [FD03] ont proposé l'un des premiers TRNG totalement implantable au sein d'un circuit FPGA. Depuis, un grand nombre générateurs a été présenté dans la littérature. Un des objectifs de cette thèse a été de proposer une méthodologie objective analysant la qualité des TRNG sélectionnés au sein de circuits reconfigurables.

La méthode d'évaluation, rapide et efficace est basée sur l'utilisation de tests statistiques matériels proposés par l'AIS 31 [AIS01]. L'intérêt de la méthode est d'évaluer les TRNG au sein du même circuit FPGA afin d'analyser ceux-ci dans des conditions similaires d'utilisation. La version matérielle de l'AIS 31 permet d'évaluer des TRNG présentant un débit maximal de 84.2 Mbits/s. Le coût matériel de celle-ci est de 4042 LUT, soit environ 10% d'un FPGA Xilinx Virtex 5 SX50. La méthode proposée permet de trouver facilement le débit optimal des TRNG au sein d'un circuit reconfigurable. Ainsi, l'intérêt d'une précaractérisation du fonctionnement d'un TRNG est prouvé.

À partir de l'étude précédente, un TRNG est sélectionné pour ses propriétés statistiques. Une architecture de générateur hybride permettant de produire des nombres aléatoires uniformes à très haut débit est ensuite proposée. Cependant, malgré la qualité du générateur sélectionné, l'aléa en sortie du TRNG peut varier au cours du temps à cause d'attaques externes possibles sur le circuit, de la variation des conditions extérieures ou encore du vieillissement des composants. La qualité aléatoire de la séquence de germes produite par un TRNG est primordiale pour assurer l'imprédictibilité d'un PRNG. Afin d'améliorer la robustesse d'un TRNG, des mécanismes de tests statistiques en ligne doivent être mis en œuvre. Ainsi, une évaluation en temps réel du comportement du générateur est réalisée et les séquences défaillances générées par celui-ci sont évitées. Cette étude propose d'utiliser une implantation efficace de la batterie du FIPS 140-2. La surface négligeable du circuit au sein des FPGA actuels et les performances obtenues par le composant permettent de contrôler la qualité du TRNG sélectionné en temps réel. Un RNG hybride à très aut débit implanté sur une cible reconfigurable et au sein d'un circuit VLSI est proposé. L'étude menée sur le RNG hybride a débouché sur un partenariat industriel réalisé avec une entreprise spécialisée en dans le domaine de la cryptographie à travers le projet pôle images et réseaux SPRING (*Shelf Proof Random Integrated Number Generator*). Dans certaines applications, il est parfois nécessaire de générer des nombres aléatoires suivant une distribution de probabilité différente d'une loi uniforme. À partir de l'utilisation du RNG hybride proposé, l'implantation d'un générateur de nombres aléatoires gaussiens à très haut débit est présenté. Par ailleurs, le circuit implanté permet de générer efficacement des nombres aléatoires suivant une loi de Rayleigh.

Après sélection d'un excellent TRNG, un PRNG à débit très élevé est recherché. La structure parallèle des automates cellulaires fait de ces PRNG une architecture très intéressante en vue d'une implantation matérielle. La seconde partie de la thèse est consacrée à la recherche de la règle optimale d'un automate cellulaire à cinq voisins. Le chapitre étudie, entre autres, le coût matériel engendré par l'implantation de tests statistiques très performants.

MOTS CLÉS : générateur vraiment aléatoire, générateur pseudo-aléatoire, tests statistiques matériels, générateur gaussien, générateur de Rayleigh, FPGA, ASIC.

Abstract

The purpose of a random number generator (RNG) is to produce a random binary sequence independent, unpredictable and uniformly distributed. A RNG is needed in all kinds of applications such as cryptography, telecommunications, computer simulations, testing of VLSI circuits or probabilistic algorithms. The number of embedded applications requiring hardware RNG is constantly increasing, especially in the reconfigurable circuits. In view of the physical design of a RNG, the solution is to use a hybrid generator composed of a truly random generator (TRNG) and a pseudo-random number generator (PRNG). Prove mathematically that a random generator is usually difficult or impossible. RNGs are often evaluated using statistical tests. Evaluation of a TRNG is much more complex than PRNG testing as the TRNG quality is a function of the noise source used in the external environment and the implementation quality within the target circuit. Fischer et al. have proposed one of the first TRNG totally embedded within an FPGA. Since, a large generator number has been presented in the literature. One objective of this thesis was to propose an objective methodology for analyzing the quality of recognized TRNGs in reconfigurable circuits.
The evaluation method, fast and efficient is based on the use of hardware statistical tests proposed by the AIS 31 and the FIPS 140-2. The advantage of the method is to evaluate the TRNGs in the same FPGA chip to analyze them in similar conditions of use. The hardware version of AIS 31 allows the TRNG evaluation with a maximum throughput of 84.2 Mb/s. The material cost of it is 4042 LUTs, about 10 % of a Xilinx Virtex 5 FPGA SX50. Moreover, the method is used to find the optimal TRNG data rate for the targeted reconfigurable circuit, showing the TRNG pre-characterization interest.

From the previous study, a TRNG is selected for its statistical properties. A high data rate hybrid generator is then proposed. However, despite the quality of the selected generator, the TRNG randomness can vary on time due to possible external attacks on the

circuit, changes in external conditions or aging components. The quality of the seed sequences provided by a TRNG is important to ensure the unpredictability of a PRNG. To improve the robustness of a TRNG, online statistical tests must be implemented. Thus, a real-time evaluation of the generator behavior is performed and the TRNG weaknesses are filtered. This study proposes to use an efficient implementation of the FIPS 140-2 battery. The negligible area and the performance allow to control the TRNG quality in real time. A hybrid RNG implemented on a reconfigurable target and within a VLSI circuit is then proposed. The study conducted on the RNG hybrid resulted in an industry partnership. Then, in some applications, it is sometimes necessary to generate random numbers following a probability distribution different from a uniform distribution. From the use of the proposed hybrid RNG, a very high data rate gaussian random number generator is proposed. Furthermore, the circuit can generate random numbers according to a Rayleigh law.

From previous studies, a hybrid RNG is proposed for FPGA implementation and is embedded into the cryptography solution proposed by a French company. The RNG generates uniform random bits and his innovating by the TRNG real time monitoring realized by embedded the FIPS 140-2 tests. Then, the realization of two ASIC in 130 nm technology are described. The chips will allow to evaluate the performance of ASIC TRNGs. Moreover, the second circuit is embedded high throughput gaussian and Rayleigh noise generators.

KEYWORDS : Truly random number generator, pseudo-random number generator, hardware statistical tests, gaussian generator, Rayleigh generator, FPGA, ASIC

VU :

Le Directeur de Thèse

Olivier SENTIEYS

VU :

Le Responsable de l'École Doctorale

VU pour autorisation de soutenance

Rennes, le

Le Président de l'Université de Rennes 1

Guy CATHELINEAU

VU après autorisation pour autorisation de publication :

Le Président de Jury,

Viktor FISCHER